L 12

```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS              YYY         YYY  SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
SSS                 YYY   YYY     SSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
   SSSSSSSSS           YYY           SSSSSSSSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
            SSS        YYY                    SSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS
```

```
_$
Ps
--
YZ
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
Z$
```

```
**FILE**ID**SYSDEFMP                                  C  3



 SSSSSSSS   YY      YY   SSSSSSSS   DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF  MM      MM  PPPPPPPP
 SSSSSSSS   YY      YY   SSSSSSSS   DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF  MM      MM  PPPPPPPP
SS          YY      YY  SS          DD      DD  EE          FF          MMMM  MMMM  PP      PP
SS          YY      YY  SS          DD      DD  EE          FF          MMMM  MMMM  PP      PP
SS            YY  YY    SS          DD      DD  EE          FF          MM  MM  MM  PP      PP
SS            YY  YY    SS          DD      DD  EE          FF          MM  MM  MM  PP      PP
 SSSSSS         YY       SSSSSS     DD      DD  EEEEEEEE    FFFFFFFF    MM      MM  PPPPPPPP
 SSSSSS         YY       SSSSSS     DD      DD  EEEEEEEE    FFFFFFFF    MM      MM  PPPPPPPP
        SS      YY              SS  DD      DD  EE          FF          MM      MM  PP
        SS      YY              SS  DD      DD  EE          FF          MM      MM  PP
        SS      YY              SS  DD      DD  EE          FF          MM      MM  PP              ....
        SS      YY              SS  DD      DD  EE          FF          MM      MM  PP              ....
SSSSSSSS        YY      SSSSSSSS    DDDDDDDD    EEEEEEEEEE  FF          MM      MM  PP              ....
SSSSSSSS        YY      SSSSSSSS    DDDDDDDD    EEEEEEEEEE  FF          MM      MM  PP              ....


 SSSSSSSS   DDDDDDDD    LL
 SSSSSSSS   DDDDDDDD    LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
SS          DD      DD  LL
 SSSSSS     DD      DD  LL
 SSSSSS     DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
        SS  DD      DD  LL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
SSSSSSSS    DDDDDDDD    LLLLLLLLLL
```

```
SYS

{+
{ D
{ u
{-
mod

/*


agg








/*




/*

end

end
```

```
{
{ Version:      'V04-000'
{
{****************************************************************************
{*                                                                          *
{*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
{*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
{*  ALL RIGHTS RESERVED.                                                    *
{*                                                                          *
{*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
{*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
{*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
{*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
{*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
{*  TRANSFERRED.                                                            *
{*                                                                          *
{*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
{*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
{*  CORPORATION.                                                            *
{*                                                                          *
{*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
{*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
{*                                                                          *
{*                                                                          *
{****************************************************************************

{++
{
{ FACILITY:  VAX/VMS System Macro Libraries
{
{ ABSTRACT:
{
{       This file contains the SDL soucre for all operating system control
{       blocks, from M to P.  That is, all control blocks from MAA to PZZ.
{
{ ENVIRONMENT:
{
{       n/a
{
{--
{
{
{ AUTHOR:  The VMS Group          CREATION DATE:  1-Aug-1976
{
{ MODIFIED BY:
{
{       V03-101 LJK0288         Lawrence J. Kenah       9-Aug-1984
{               Add AUTHPRI cell to PCB that duplicates existing PHD field.
{
{       V03-100 ACG0440         Andrew C. Goldstein,    23-Jul-1984  11:26
{               Add ref count and classification valid flag to ORB
{
{       V03-099 ROW0397         Ralph O. Weber          21-JUL-1984
{               Add definition for "position lost" MSCP end flag
{
{       V03-098 ROW0374         Ralph O. Weber          19-JUL-1984
```

```
SYS

mod
/**
/*
/*
/*
/*
/*-

agg






end

end
```

```
{               Add an entry to the MVMSL at a negative offset from the MVMSL
{               base which gives the maximum MVMSL index value, MVMSL$B_MAXIDX.
{               Add the following MSCP controller error subcodes;
{               MSCP$K_SC_EDCER, MSCP$K_SC_DTSTR, and MSCP$K_SC_REMRSRC.  Add
{               symbol definitions for the three kind of shadow copy operation;
{               MSCP$K_CS_NOCPY, MSCP$K_CS_COPY, and MSCP$K_CS_MGCPY.  Add a
{               no members subcode for MSCP available, MSCP$K_SC_NOMEMB.  Add
{               definitions related to bad block replacement error logging.
{
{       V03-097 RLRBINDT1         Robert L. Rappaport      12-Jul-1984
{               Add more BI devices to $NDTDEF.
{
{       V03-096 LMP0271           L. Mark Pilant,          29-Jun-1984  13:03
{               Add ORB$V_NOACL to indicate that the object can not
{               have an ACL.
{
{       V03-095 WMC0095           Wayne Cardoza            02-May-1984
{               Add PHD$M_NO_WS_CHNG
{
{       V03-094 NPK3051           N. Kronenberg            17-Apr-1984
{               Add MSCP$K_EMD_EMUL to $MSCPDEF.
{
{       V03-093 GRR3093           Gregory R. Robert        11-Apr-1984
{               Added $PSMDEF (previously part of $SMBDEF)
{
{       V03-092 ACG0415           Andrew C. Goldstein,     11-Apr-1984  15:18
{               Update $NSAARGDEF for last set of audit changes
{
{       V03-091 RLRPDTADP         Robert L. Rappaport      10-Apr-1984
{               Add PDT$L_ADP to common $PDTDEF.
{
{       V03-090 SSA0024           Stan Amway               10-Apr-1984
{               Add PIB$B_SRQ_ACCESS and PIB$W_SRQ_ACON to module $PIBDEF.
{
{       V03-089 ROW0341           Ralph O. Weber           9-APR-1984
{               Correct equated value for MSCP$K_CL_D144, an MSCP controller
{               or unit identifier class.  Add MVMSL$M_SUPRESS, a flag to
{               indicate that a given mount verification message can be
{               suppressed.
{
{       V03-088 WHM0002           Bill Matthews            09-Apr-1984
{               Added additional size constants in $PRMDEF to support
{               long ascii sysgen parameters.
{
{       V03-087 LMP0221           L. Mark Pilant,          9-Apr-1984  12:16
{               Add additional subfields to the ORB definition.
{
{       V03-086 MHB0132           Mark Bramhall            9-Apr-1984
{               Add SPAWN_CLI and SPAWN_TABLE to $PQBDEF.
{
{       V03-085 NPK3048           N. Kronenberg            5-Apr-1984
{               Add protocol level and vc failure reason to $PBDEF.
{
{       V03-084 KPL0001           Peter Lieberwirth        22-Mar-1984
{               Add $NBIADEF.
{
```

```
SYS

mod
/*+
/*
/*-

agg




end

end
```

```
{       V03-083 NPK3047         N. Kronenberg           22-Mar-1984
{               Change 1st longword in PDT from reserved to forward
{               link.  Add a new port driver vector to $PDTDEF,
{               PDT$L_STOP_VCS.
{
{       V03-082 RLRPDTUCBO      Robert L. Rappaport     21-Mar-1984
{               Add PDT$L_UCBO field to common PDT.
{
{       V03-081 LMP0214         L. Mark Pilant,         21-Mar-1984  9:48
{               Add ORB$L_ACL_MUTEX, the ACL mutex to the ORB.
{
{       V03-080 SSA0019         Stan Amway              13-Mar-1984
{               In $PMBDEF, add ACMODE field to indicate owner access mode.
{
{       V03-079 LMP0206         L. Mark Pilant,         9-Mar-1984  14:27
{               Add a new structure, $ORBDEF, to define an Object's Rights
{               Block.
{
{       V03-078 NPK3046         N. Kronenberg           7-Mar-1984
{               Add PDT$L_POLLSWEEP to PDTDEF.  This field contains
{               a port driver estimate of the number of seconds it
{               will take to discover all possible ports currently
{               in the cluster.
{               Add PBO$K_LENGTH to PBODEF.
{
{       V03-077 SSA0012         Stan Amway              27-Feb-1984
{               In $PMBDEF, move overflow counter out of ACB_OVERLAY.
{               Use former location of overflow counter as address of
{               piggy-back kernel AST routine.
{
{               Add flag PMB$V_QAST to indicate imbedded ACB is queued.
{
{       V03-076 ROW0315         Ralph O. Weber          27-FEB-1984
{               Add new controller models produced by revised MSCP
{               specification to $MSCPDEF.
{
{       V03-075 ROW0314         Ralph O. Weber          27-FEB-1984
{               Add $MVMSLDEF, mount verification messages list, structure
{               definition.
{
{       V03-074 ROW0313         Ralph O. Weber          27-FEB-1984
{               Correct MSCP$K_ST_RDTRN, MSCP$K_ST_PLOST, MSCP$K_ST_PRESE, and
{               MSCP$K_ST_LED definitions (they all had values one less than
{               they should be).
{
{       V03-073 MMD0243         Meg Dumont,     24-Feb-1984  14:56
{               Add support for MVL$V_OPER adn MVL$B_STDVER
{
{       V03-072 SSA0008         Stan Amway              10-Feb-1984
{               Changed timestamp in $PIBDEF to a quadword.
{
{       V03-071 SSA0007         Stan Amway              6-Feb-1984
{               Modified $PFBDEF and $PMBDEF to track changes
{               in page fault monitor buffer management routines.
{
{       V03-070 RSH0096         R. Scott Hanna          02-Feb-1984
{               Add mandatory security auditing bit to PCB status bits.
{               Replace $NSAARGDEF, $NSAEVTDEF, and $NSAIDTDEF.
{
{       V03-069 WHM0001         Bill Matthews           01-Feb-1984
{               Add a new parameter type for the LGI_ SYSGEN parameters.
{
{       V03-068 TMK0002         Todd M. Katz            31-Jan-1984
{               Increase the number of reserved vectors in $PDTDEF from 3 to 10.
{
{       V03-067 ROW0291         Ralph O. Weber          29-JAN-1984
{               Add MSCP unit number definitions for foreign disk "old
{               controller sub types."  This plus the 8 foreign disk device
{               types (DT$_FD1 through DT$_FD8) allow foreign disks to be
{               served using the MSCP server.
{
{       V03-066 ACG0385         Andrew C. Goldstein,    10-Jan-1984  16:21
{               New network proxy file format ($NAFDEF)
{
{       V03-064 LJK0257         Lawrence J. kenah       5-Jan-1984
{               Increase all text fields in PQB to 256 characters to
{               accommodate longer logical names, file specifications, and
{               so on. Move page file parameters and SWAPSIZE from PQB to
{               PCB to allow PQB to be allocated from paged memory. Add
{               security clearance fields to PHD.
{
{       V03-063 ROW0271         Ralph O. Weber          29-DEC-1983
{               Add MSCP structure definitions for shadowed volume set
{               operations to $MSCPDEF.  The unit flags bitfield definitions
{               have been omitted from this update because their exact values
{               are, as yet, unclear.
{
{       V03-062 SSA0004         Stan Amway              28-Dec-1983
{               Added support for page fault monitor enhancements.
{               Added PCB$L_PMB to PCB.
{               Replaced module $PMBDEF with a substantially changed version.
{               Added new module $PFBDEF.
{
{       V03-061 ROW0264         Ralph O. Weber          27-DEC-1983
{               Replace the entire $MSCPDEF module with a new, more readable
{               version.  Several previously undefined mask symbols are now
{               defined.  The new $MSCPDEF is believed to produce useable
{               symbols for BLISS.
{
{       V03-060 TCM0001         Trudy C. Matthews       14-Dec-1983
{               Add new nexus device type code: NDT$_MEM1664NI, to describe
{               11/750 memory controller, which can hold a mixture of
{               16k and 64k chip array cards.
{
{       V03-059 LMP0177         L. Mark Pilant,         7-Dec-1983  9:51
{               Add an ACL listhead to the PCB.
{
{       V03-058 SSA0003         Stan Amway              5-Dec-1983
{               Added support for outswap scheduling changes.
{               Removed PHD$W_WAITIME from PHD.
{               Added   PCB$L_WAITIME to   PCB.
{
```

```
{       V03-057 WMC0057         Wayne Cardoza           05-Dec-1983
{               Change PHD$W_BAK,_WSLX,PSTBASMAX to longwords.
{
{       V03-056 JWT0145         Jim Teague              18-Nov-1983
{               Define masks for $PCBDEF bitfields.
{
{       V03-055 ROW0249         Ralph O. Weber          10-NOV-1983
{               Add MSCP$B_CNT_ALCS, the MSCP Set Controller Characteristics
{               end message field in which the allocation class will be
{               returned.
{
{       V03-054 LMP0167         L. Mark Pilant,         10-Nov-1983  15:22
{               Modify $NMBDEF to add support for full ODS-1 wildcarding.
{
{       V03-053 RLRBINDT        Robert L. Rappaport     9-Nov-1983
{               Add BI devices to $NDTDEF.
{
{       V03-052 TMK0001         Todd M. Katz            26-Oct-1983
{               Add PQB$L_JTQUOTA to $PQBDEF.
{
{       V03-051 SSA00002        Stan Amway              30-Sep-1983
{               Module $PIBDEF - added aggregate PIBDEF5 to support
{               new routine PMS$ABORT_RQ in IOPERFORM. Added aborted
{               I/O request as new transaction type in aggregate
{               PIBDEF4.
{
{       V03-050 KTA3084         Kerbey T. Altmann       27-Sep-1983
{               Differentiate between RP/RM on MASSBUS disks in
{               the MSCP unit number.
{
{       V03-049 SSA0001         Stan Amway              13-Sep-1983
{               Module $PIBDEF, aggregate PIBDEF2 - Made transfer
{               byte count a longword and relocated to end of
{               structure. Added FILL_9 to redefine word formerly
{               used as transfer byte count.
{
{       V03-048 WMC0047         Wayne Cardoza           31-Aug-1983
{               Add a flag to MMGDEF
{
{       V03-047 GAS0171         Gerry Smith             24-Aug-1983
{               For NSAEVTDEF, remove terminal and mailbox I/O, and
{               add interactive and remote login/logout.
{
{       V03-046 WMC0046         Wayne Cardoza   28-Jul-1983
{               Add PQB fields for logical name characteristics.
{
{       V03-045 RSH0048         R. Scott Hanna          28-Jul-1983
{               Replace $NSAARGDEF
{
{       V03-044 RLRTMSCP        Robert L. Rappaport     28-Jul-1983
{               Add MSCP$K_ST_LED (LEOT detected status),
{               MSCP$K_SC_DLATE( Data Late subcode), MSCP$x_MD_IMMED
{               (request Immediate completion modifier) and MSCP$x_MD_DLEOT
{               (request LEOT detection modifier).
{
{       V03-043 NPK3029         N. Kronenberg           26-Jul-1983
```

```
SYS

end

end
```

```
{               Rearrange PB slightly.  Add new send dg w/register
{               entry to PDT and 3 new reserved longwords at end
{               of SCS entry list.
{
{       V03-042 RNG0042         Rod N. Gamache          22-Jul-1983
{               Add MS780-H nexus device types to $NDTDEF.
{
{       V03-041 RLRMSLG         Robert L. Rappaport     22-Jul-1983
{               Add MSLG (MSCP Error Log) definitions.
{               Also add "Host Buffer Access Error" sub-codes to MSCPDEF.
{
{       V03-040 MSH0002         Maryann Hinden          23-Jun-1983
{               Add $PRCPOLDEF.
{
{       V03-039 RSH0037         R. Scott Hanna          17-Jun-1983
{               Permanent fix to $NSAARGDEF symbols. Add ARG_COUNT
{               to $NSAARGDEF. Add EVT_UPGRADE and EVT_DOWNGRADE to
{               $NSAEVTDEF.
{
{       V03-038 SRB0093         Steve Beckhardt         6-Jun-1983
{               Added PCB$M_RECOVER to $PCBDEF
{
{       V03-037 ADE9001         A. ELDRIDGE             27-May-1983
{               Temporary fix to $NSAARGDEF to allow system to build.
{
{       V03-036 RSH0024         R. Scott Hanna          24-May-1983
{               Add $NSAIDTDEF and $NSAARGDEF
{
{       V03-035 RLRPCHAR        Robert L. Rappaport     19-May-1983
{               Add PDT$W_PORTCHAR field and add PDT$M_SNGLHOST bit
{               in this word.
{
{       V03-034 KTA3051         Kerbey T. Altmann       18-May-1983
{               Add more PDT types.
{
{       V03-033 LMP0112         L. Mark Pilant,         10-May-1983  9:24
{               Add a new cell, PCB$L_DEFPROT, to contain the process
{               default protection.
{
{       V03-032 RSH0016         R. Scott Hanna          30-Apr-1983
{               Replace $NSAEVTDEF
{
{       V03-031 MSH0001         Maryann Hinden          25-Mar-1983
{               Add ASCII type flag to $PRMDEF
{
{       V03-030 MMD0110         Meg Dumont,     24-Mar-1983  17:53
{               Fix the def for MVL$K(C)_FIXLEN
{
{       V03-029 WMC0029         Wayne Cardoza           15-Mar-1983
{               Add IMGDMP flag to PHD
{               Add flags word to PQB.
{
{       V03-028 RSH0011         R. Scott Hanna          13-Mar-1983
{               Add $NSAEVTDEF
{
{       V03-027 MMD0107         Meg Dumont,     10-Mar-1983  15:54
{               Add field MVL$T_VOLOWNER to contain VOL1 owner id field
{
{       V03-026 RLRUNIT         Robert L. Rappaport     8-Mar-1983
{               Add subfields to MSCP$W_UNIT for HSC emulator.
{
{       V03-025 WMC0025         Wayne Cardoza           07-Mar-1983
{               Add PCB$V_INTER
{
{       V03-024 RLRDDB          Robert L. Rappaport     1-Mar-1983
{               Add subfields to MSCP$L_MEDIA_ID.
{
{       V03-023 DWT0079         David W. Thiel          1-Mar-1983
{               Add PRM$V_CLUSTER to define cluster SYSGEN parameters.
{
{       V03-022 RLRMXBCNT       Robert L. Rappaport     25-Feb-1983
{               Add PDT$L_MAXBCNT.
{
{       V03-021 CWH1002         CW Hobbs                24-Feb-1983
{               Rename PCB$L_PID_EXTERNAL to PCB$L_EPID, and add PCB$L_EOWNER.
{
{       V03-020 KBT0499         Keith B. Thompson       16-Feb-1983
{               Increase the size of PQB$C_MAXDIRLEN to 178 (match FWA$C_MAXDIRLEN)
{
{       V03-019 CWH0001         CW Hobbs                19-Feb-1983
{               Add PCB$L_PID_EXTERNAL for pid changes.
{
{       V03-018 RLRRDRX         Robert L. Rappaport     9-Feb-1983
{               Add MSCP$K_CM_RDRX, MSCP$K_OP_RWATN and MSCP$M_MD_EXCLU.
{
{       V03-017 WMC0016         Wayne Cardoza           26-Jan-1983
{               Make PTE$V_STX a signed bitfield.
{
{       V03-016 ACG0307         Andrew C. Goldstein,    10-Jan-1983  16:25
{               Add system protection block ($PRBDEF)
{
{       V03-015 WMC0015         Wayne Cardoza           10-Jan-1982
{               Put PRMDEF back.
{               It was accidentally deleted in V03-013.
{
{       V03-014 WMC0014         Wayne Cardoza           8-Jan-1983
{               Temporarily delete PRODEF to make build run.
{               There is a conflict with the object language.
{
{       V03-013 ACG0307         Andrew C. Goldstein,    30-Dec-1982  17:08
{               Add enhanced protection structures to PCB
{
{       V03-012 ACG0303         Andrew C. Goldstein,    9-Dec-1982  15:13
{               Add FILL attribute to extraneous names
{
{       V03-011 NPK3010         N. Kronenberg           12-Nov-1982
{               Add $PBODEF to define offsets to output from SCS$CONFIG_PTH
{               call.
{               Add CI port type codes symbols to $PBDEF.
{
{       V03-010 CDS0001         C Saether               22-Oct-1982
{               Add PCB$B_DPC delete pending counter.
```

```
{
{       V03-009 RLRPRESE        Robert L. Rappaport     15-Oct-1982
{               Add MSCP$K_ST_PRESE (previously existing serious exception).
{
{        V03-008        WMC0002                 Wayne Cardoza           28-Sep-1982
{               Expand PQB to add page file selection.
{
{       V03-007         WMC0001                 Wayne Cardoza                   28-Jul-1982
{               Add "useable by checkpoint" bit to page file control block.
{
{       V03-006         RLR0002                 Robert L. Rappaport             13-July-1982
{               Correct some Tape MSCP errors.
{
{       V03-005         RLR0001                 Robert L. Rappaport             17-June-1982
{               Add Tape MSCP definitions to MSCPDEF.
{
{
{**
```

```
module $MBADEF;

/*+
/* MASSBUS ADAPTER REGISTER OFFSET DEFINITIONS
/*-


aggregate MBADEF structure prefix MBA$;
    CSR_OVERLAY union fill;
        CSR longword unsigned;                          /*CONFIGURATION STATUS REGISTER
        CSR_BITS structure fill;
            CSR_ADCOD bitfield length 8;                /* ADAPTER CODE FIELD
            FILL_1 bitfield length 13 fill prefix MBADEF tag $$;/* RESERVED BITS
            CSR_OT bitfield mask;                       /* OVER TEMPERATURE
            CSR_PU bitfield mask;                       /* ADAPTER POWER UP
            CSR_PD bitfield mask;                       /* ADAPTER POWER DOWN
            FILL_2 bitfield length 2 fill prefix MBADEF tag $$;/* RESERVED BITS
            CSR_XMFLT bitfield mask;                    /* TRANSMITTER FAULT
            CSR_MT bitfield mask;                       /* MULTIPLE TRANSMITTERS
            FILL_3 bitfield fill prefix MBADEF tag $$;  /* RESERVED BIT
            CSR_URD bitfield mask;                      /* UNEXPECTED READ DATA
            CSR_WS bitfield mask;                       /* WRITE SEQUENCE DATA
            CSR_PE bitfield mask;                       /* SBI PARITY ERROR
        end CSR_BITS;
    end CSR_OVERLAY;
    CR_OVERLAY union fill;
        CR longword unsigned;                           /*CONTROL REGISTER
        CR_BITS structure fill;
            CR_INIT bitfield mask;                      /* ADAPTER INITIALIZATION
            CR_ABORT bitfield mask;                     /* ABORT OPERATION
            CR_IE bitfield mask;                        /* INTERRUPT ENABLE
        end CR_BITS;
    end CR_OVERLAY;
    SR_OVERLAY union fill;
        SR longword unsigned;                           /*STATUS REGISTER
        SR_BITS structure fill;
            SR_RDTO bitfield mask;                      /* READ DATA TIMEOUT
            SR_ISTO bitfield mask;                      /* INTERFACE SEQUENCE TIMEOUT
            SR_RDS bitfield mask;                       /* READ DATA SUBSTITUTE
            SR_ERCONF bitfield mask;                    /* ERROR CONFIRMATION
            SR_INVMAP bitfield mask;                    /* INVALID MAP REGISTER
            SR_MAPPE bitfield mask;                     /* MAP PARITY ERROR
            SR_MDPE bitfield mask;                      /* MASSBUS DATA PARITY ERROR
            SR_MBEXC bitfield mask;                     /* MASSBUS EXCEPTION
            SR_MXF bitfield mask;                       /* MISSED TRANSFER ERROR
            SR_WCKLWR bitfield mask;                    /* WRITE CHECK ERROR LOWER BYTE
            SR_WCKUPR bitfield mask;                    /* WRITE CHECK ERROR UPPER BYTE
            SR_DLT bitfield mask;                       /* DATA LATE ERROR
            SR_DTABT bitfield mask;                     /* DATA TRANSFER ABORTED
            SR_DTCOMP bitfield mask;                    /* DATA TRANSFER COMPLETE
            SR_SPE bitfield mask;                       /* SILO PARITY ERROR
            FILL_4 bitfield fill prefix MBADEF tag $$;  /* RESERVED BITS
            SR_ATTN bitfield mask;                      /* MASSBUS ATTENTION
            SR_MCPE bitfield mask;                      /* MASSBUS COMTROL PARITY ERROR
            SR_NED bitfield mask;                       /* NONEXISTENT DRIVE
            SR_PGE bitfield mask;                             /* PROGRAM ERROR
            FILL_5 bitfield length 3 fill prefix MBADEF tag $$;/* RESERVED BITS
            SR_CBHUNG bitfield mask;                          /* CB HUNG
            FILL_6 bitfield length 5 fill prefix MBADEF tag $$;/* RESERVED BITS
            SR_CRD bitfield mask;                             /* CORRECTED READ DATA
            SR_NRCONF bitfield mask;                          /* NO RESPONSE CONFIRMATION
            SR_DTBUSY bitfield mask;                          /* DATA TRANSFER BUSY
        end SR_BITS;
                                                              /* ERROR BITS
        constant ERROR  equals
            ( MBA$M_SR_RDTO!
                MBA$M_SR_ISTO!
                MBA$M_SR_RDS!
                MBA$M_SR_ERCONF!
                MBA$M_SR_INVMAP!
                MBA$M_SR_MAPPE!
                MBA$M_SR_MDPE!
                MBA$M_SR_MBEXC!
                MBA$M_SR_MXF!
                MBA$M_SR_WCKLWR!
                MBA$M_SR_WCKUPR!
                MBA$M_SR_DLT!
                MBA$M_SR_SPE!
                MBA$M_SR_DTABT!
                MBA$M_SR_MCPE!
                MBA$M_SR_NED!
                MBA$M_SR_PGE  )
          prefix MBA tag $M;                                  /* PROGRAM ERROR
    end SR_OVERLAY;
    VAR longword unsigned;                                    /*VIRTUAL ADDRESS REGISTER
    BCR longword unsigned;                                    /*BYTE COUNT REGISTER
    DR longword unsigned;                                     /*DIAGNOSTIC REGISTER
    SELMR longword unsigned;                                  /*SELECTED MAP REGISTER
    FILL_7 byte dimension 996 fill prefix MBADEF tag $$;/* VALUE IS 1024-<4*7>
    ERB_OVERLAY union fill;
        ERB longword unsigned;                                /*BASE ADDRESS OF EXTERNAL REGISTERS
        ERB_BITS structure fill;
            FILL_8 bitfield length 7 fill prefix MBADEF tag $$;/* REGISTER OFFSET ADDRESS BITS
            ERB_UNIT bitfield length 3;                       /* DRIVE UNIT NUMBER
        end ERB_BITS;
    end ERB_OVERLAY;
    FILL_9 byte dimension 12 fill prefix MBADEF tag $$; /* DRIVE REGISTER ADDRESS SPACE
    AS longword unsigned;                                     /*ATTENTION SUMMARY REGISTER
    FILL_10 byte dimension 1004 fill prefix MBADEF tag $$;/* VALUE IS 2048-.
                                                              /* TO POSITION TO 2048
    MAP longword unsigned dimension 256;                      /*MAP REGISTERS

end MBADEF;

end_module $MBADEF;
```

```
module $MBXDEF;
/*+
/* SHARED MEMORY MAILBOX CONTROL BLOCK DEFINITIONS
/*
/* THERE IS ONE MAILBOX CONTROL BLOCK FOR EACH MAILBOX IN SHARED
/* MEMORY.  ANY PROCESSOR THAT WANTS TO ACCESS THE MAILBOX CREATES
/* A UCB TO CONTROL ACCESS TO THE MAILBOX.
/*-


aggregate MBXDEF structure prefix MBX$;
    MSG quadword unsigned;                              /*MESSAGE QUEUE LISTHEAD
    FLAGS_OVERLAY union fill;
        FLAGS byte unsigned;                            /*FLAGS
        FLAGS_BITS structure fill;
            ALLOC bitfield mask;                        /* MAILBOX ALLOCATED
            VALID bitfield mask;                        /* MAILBOX INITIALIZED AND USEABLE
            DELPEND bitfield mask;                      /* DELETE PENDING
            QUOTALCK bitfield mask;                     /* QUOTA/COUNT MODIFICATION LOCK
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    CREATPORT byte unsigned;                            /*PORT NUMBER OF MAILBOX CREATOR
    UNIT word unsigned;                                 /*MAILBOX UNIT NUMBER
    "REF" word unsigned;                                /*REFERENCE FLAGS (1 BIT/PORT)
    READER word unsigned;                               /*WAITING READER (1 BIT/PORT)
    READAST word unsigned;                              /*WAITING READ AST (1 BIT/PORT)
    WRITAST word unsigned;                              /*WAITING WRITE AST (1 BIT/PORT)
    MAXMSG word unsigned;                               /*MAXIMUM MESSAGE SIZE
    MSGCNT word unsigned;                               /*CURRENT NUMBER OF MESSAGES
    BUFFQUO word unsigned;                              /*BUFFER QUOTA
    PROT word unsigned;                                 /*PROTECTION MASK
    OWNUIC longword unsigned;                           /*OWNER UIC
    NAME character length 16;                           /*MAILBOX NAME (COUNTED STRING)
/* *** THE LENGTH OF THIS STRUCTURE MUST BE AN EVEN MULTIPLE OF 8 ***
/* *** BECAUSE THE MESSAGE QUEUE HEADER MUST BE QUADWORD ALIGNED ***
    constant "LENGTH" equals . prefix MBX$ tag K;       /*LENGTH OF STRUCTURE
    constant "LENGTH" equals . prefix MBX$ tag C;       /*LENGTH OF STRUCTURE

end MBXDEF;

end_module $MBXDEF;
```

```
module $MCHKDEF;
/*+
/* MACHINE CHECK ERROR RECOVERY BLOCK MASK BIT DEFFINITIONS
/* BITS USED TO FILTER AND TEST FOR ERROR TYPES
/*-

aggregate MCHKDEF  union prefix MCHK$;
    MCHKDEF_BITS structure fill;
        LOG bitfield mask;                                  /*INHIBIT ERROR LOGGIN FOR THE ERROR
        MCK bitfield mask;                                  /*PROTECT AGAINST MACHINE CHECKS
        NEXM bitfield mask;                                 /*PROTECT AGAINST NON-EXISTENT MEMORY
        UBA bitfield mask;                                  /*PROTECT AGAINST UBA ADAPTER ERROR INTRPT
    end MCHKDEF_BITS;
end MCHKDEF;

end_module $MCHKDEF;
```

```
{+
{ Define the frame pointer offsets that determine what the impure area
{ used by the memory management system services looks like.
{-
module $MMGDEF;

/*         -F          ,B,0                          /* ending address of negated structure
                                                              /* (needed to obtain length definition)


aggregate MMGDEF structure prefix MMG$ origin FILL_2;
    constant 'LENGTH' equals . prefix MMG$ tag K;       /* size of scratch area
    constant 'LENGTH' equals . prefix MMG$ tag C;       /* size of scratch area
    EFBLK longword unsigned;                            /* stored end-of-file block from WCB
    VFYFLAGS longword unsigned;                         /* verified section flags and maximum
                                                        /*  access mode for writing
    SVSTARTVA longword unsigned;                        /* saved starting virtual address
    PAGESUBR longword unsigned;                         /* address of per page subroutine
    SAVRETADR longword unsigned;                        /* saved return address range
    CALLEDIPL longword unsigned;                        /* caller's IPL
    MAXACMODE_OVERLAY union fill;
        MAXACMODE longword unsigned;                    /* maximized read access mode
/*
        MAXACMODE_BITS structure fill;
            FILL_1 bitfield length 8 fill prefix MMGDEF tag $$;/* no flags in first byte
            CHGPAGFIL bitfield mask;                    /* charge page file for this PTE
            DELGBLDON bitfield mask;                    /* global pages in this range
                                                        /*  already deleted
            NOWAIT_IPLO bitfield mask;                  /* abort instead of dropping to 0
        end MAXACMODE_BITS;
/*
    end MAXACMODE_OVERLAY;
    FILL_2 byte fill prefix MMGDEF tag $$;
end MMGDEF;

end_module $MMGDEF;
```

```
module $MTLDEF;
/*+
/* MOUNTED VOLUME LIST ENTRY. ONE SUCH ENTRY APPEARS IN THE PROCESS MOUNTED
/* VOLUME LIST FOR EACH VOLUME MOUNTED BY THE PROCESS AS /SHARE OR /NOSHARE.
/* IN ADDITION, EACH VOLUME MOUNTED /SYSTEM OR /GROUP HAS AN ENTRY IN THE
/* SYSTEM WIDE MOUNTED VOLUME LIST.
/*-


aggregate MTLDEF structure prefix MTL$;
    MTLFL longword unsigned;                            /* FORWARD LIST POINTER
    MTLBL longword unsigned;                            /* BACK LIST POINTER
    SIZE word unsigned;                                 /* STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                                 /* STRUCTURE TYPE CODE
    STATUS_OVERLAY union fill;
        STATUS byte unsigned;                           /* STATUS BYTE
        STATUS_BITS structure fill;
            VOLSET bitfield;                            /* ENTRY IS FOR A VOLUME SET
        end STATUS_BITS;
    end STATUS_OVERLAY;
    UCB longword unsigned;                              /* POINTER TO DEVICE UCB
    LOGNAME longword unsigned;                          /* POINTER TO ASSOCIATED LOGICAL NAME
    FILL_1 longword fill prefix MTLDEF tag $$;          /* RESERVED LONGWORD
    constant "LENGTH" equals . prefix MTL$ tag K;       /* LENGTH OF STRUCTURE
    constant "LENGTH" equals . prefix MTL$ tag C;       /* LENGTH OF STRUCTURE
end MTLDEF;

end_module $MTLDEF;
```

```
module $MTXDEF;
/*+
/* MUTEX DEFINITIONS
/*-


aggregate MTXDEF union prefix MTX$;
    FILL_1 longword fill prefix MTXDEF tag $$;
        FILL_1_BITS structure fill;
            FILL_2 bitfield length 16 fill prefix MTXDEF tag $$;
            WRT bitfield;                                   /* WRITE PENDING OR IN PROGRESS
        end FILL_1_BITS;
        FILL_1_FIELDS structure fill;
            OWNCNT word unsigned;                           /*OWNERSHIP COUNT
            STS word unsigned;                              /*STATUS BITS

        end FILL_1_FIELDS;
end MTXDEF;

end_module $MTXDEF;
```

```
module $MPMDEF;
/*+
/* MULTIPORT MEMORY (MA780/MA750) ADAPTER REGISTER OFFSET DEFINITIONS
/*-
/*
/* The UETP for the MA780 depends on some of the following definitions.  Please
/* let someone in that group know if the definitions change substantially.
/*


constant PORTS  equals 4  prefix MPM tag $C;            /*MAXIMUM NUMBER OF PORTS PER MEMORY

aggregate MPMDEF structure prefix MPM$;
    CSR_OVERLAY union fill;
        CSR longword unsigned;                          /*CONFIGURATION STATUS REGISTER
        CSR_BITS0 structure fill;
            CSR_PORT bitfield mask length 2;            /* PORT NUMBER
        end CSR_BITS0;
        CSR_BITS1 structure fill;
            CSR_ADCOD bitfield mask length 8;           /* ADAPTER CODE FIELD
            FILL_1 bitfield length 14 fill prefix MPMDEF tag $$;/* RESERVED BITS
            CSR_PU bitfield mask;                       /* ADAPTER POWER UP
            CSR_PD bitfield mask;                       /* ADAPTER POWER DOWN
            FILL_2 bitfield length 2 fill prefix MPMDEF tag $$;/* RESERVED BITS
            CSR_XMFLT bitfield mask;                    /* TRANSMITTER FAULT
            CSR_MT bitfield mask;                       /* MULTIPLE TRANSMITTERS
            CSR_IS bitfield mask;                       /* INTERLOCK SEQUENCE
            FILL_3 bitfield fill prefix MPMDEF tag $$;  /* RESERVED BIT
            CSR_WS bitfield mask;                       /* WRITE SEQUENCE DATA
            CSR_PE bitfield mask;                       /* SBI PARITY ERROR
        end CSR_BITS1;
        constant CSR_TYPE       equals 64  prefix MPM tag $C;/* MULTIPORT ADAPTER TYPE CODE
    end CSR_OVERLAY;
    CR_OVERLAY union fill;
        CR longword unsigned;                           /*CONTROL REGISTER
        CR_BITS structure fill;
            CR_MIE bitfield mask;                       /* MASTER INTERRUPT ENABLE
            CR_EIE bitfield mask;                       /* ERROR INTERRUPT ENABLE
            FILL_4 bitfield length 22 fill prefix MPMDEF tag $$;/*
            CR_ERRS bitfield mask length 8;             /* PORT INTERFACE ERRORS
        end CR_BITS;
    end CR_OVERLAY;
    SR_OVERLAY union fill;
        SR longword unsigned;                           /*STATUS REGISTER
        SR_BITS structure fill;
            FILL_5 bitfield fill prefix MPMDEF tag $$;  /* (UNUSED)
            SR_EIE bitfield mask;                       /* ERROR INTERRUPT ENABLE
            FILL_6 bitfield length 11 fill prefix MPMDEF tag $$;/*
            SR_SS bitfield mask;                        /* SINGLE STEP
            SR_IDL bitfield mask;                       /* INVALIDATE DATA LOST IN MPC
            SR_IT bitfield mask;                        /* INTERLOCK TIMEOUT
            FILL_7 bitfield length 12 fill prefix MPMDEF tag $$;/*
            SR_AGP bitfield mask;                       /* ADMI GRANT PARITY ERROR
            SR_XDF bitfield mask;                       /* XMIT DURING FAULT
            SR_MXF bitfield mask;                       /* MULTIPLE XMITTER FAULT
            SR_ACA bitfield mask;                       /* ADMI COMMAND ABORT
        end SR_BITS;
    end SR_OVERLAY;
    INV_OVERLAY union fill;
        INV longword unsigned;                          /*INVALIDATION CONTROL REGISTER
        INV_BITS structure fill;
            INV_ID bitfield mask length 16;             /* CACHED DEVICE NEXUS ID'S
            INV_MEMSZ bitfield mask length 3;           /* MEMORY SIZE (256KB BOARDS)
            FILL_8 bitfield fill prefix MPMDEF tag $$;  /* (UNUSED)
            INV_STADR bitfield mask length 11;          /* STARTING SBI ADDR OF MEMORY
            INV_CACHF bitfield mask;                    /* CACHED FORCE (IGNORE ID'S)
        end INV_BITS;
    end INV_OVERLAY;
    ERR_OVERLAY union fill;
        ERR longword unsigned;                          /*ARRAY ERROR REGISTER
        ERR_BITS structure fill;
            FILL_9 bitfield length 28 fill prefix MPMDEF tag $$;/*
            ERR_ELR bitfield mask;                      /* ERROR LOG REQUEST
            ERR_HI bitfield mask;                       /* HIGH ERROR RATE
            ERR_ICRD bitfield mask;                     /* INHIBIT CRD ERRORS
            ERR_IMP bitfield mask;                      /* INVALIDATE MAP PARITY ERROR
        end ERR_BITS;
    end ERR_OVERLAY;
    CSR0_OVERLAY union fill;
        CSR0 longword unsigned;                         /*CONFIGURATION STATUS REGISTER 0
        CSR0_BITS structure fill;
            FILL_10 bitfield length 4 fill prefix MPMDEF tag $$;/*
            CSR0_POW bitfield length 4;                 /* PER PORT POWER STATUS
            CSR0_ERR bitfield length 4;                 /* PER PORT ERROR STATUS
            CSR0_ONL bitfield length 4;                 /* PER PORT ONLINE STATUS
        end CSR0_BITS;
    end CSR0_OVERLAY;
    CSR1_OVERLAY union fill;
        CSR1 longword unsigned;                         /*CONFIGURATION STATUS REGISTER 1
        CSR1_BITS structure fill;
            FILL_11 bitfield length 10 fill prefix MPMDEF tag $$;/*
            CSR1_MIA bitfield mask;                     /* MULTIPLE INTERLOCK ACCEPTED
        end CSR1_BITS;
    end CSR1_OVERLAY;
    MR_OVERLAY union fill;
        MR longword unsigned;                           /*MAINTENANCE REGISTER
        MR_BITS structure fill;
            FILL_12 bitfield length 14 fill prefix MPMDEF tag $$;/* (ERROR BITS)
            MR_UNIT bitfield length 2;                  /* MEMORY UNIT NUMBER
        end MR_BITS;
    end MR_OVERLAY;
    IIR_OVERLAY union fill;
        IIR longword unsigned;                          /*INTERPORT INTERRUPT REQUEST REGISTER
        IIR_BITS structure fill;
            IIR_STS bitfield length 16;                 /* STATUS BITS (WRITE TO CLEAR)
            IIR_CTL bitfield length 16;                 /* CONTROL BITS (WRITE TO SET STATUS BITS)
        end IIR_BITS;
    end IIR_OVERLAY;
    IIE_OVERLAY union fill;
        IIE longword unsigned;                          /*INTERPORT INTERRUPT ENABLE REGISTER
        IIE_BITS structure fill;
            IIE_STS bitfield length 16;                 /* CONTROL BITS (WRITE TO CLEAR)
```

```
SYSD
end
/* D
aggr
end
/* D
aggr
end
/* D
aggr
end
/* D
```

```
            IIE_CTL bitfield length 16;                   /* STATUS BITS (WRITE TO SET STATUS BITS)
         end IIE_BITS;

      end IIE_OVERLAY;
end MPMDEF;

end_module $MPMDEF;
```

```
module $MSLGDEF;

/*+
/* MSLG, MScp error LoG message definitions
/*      These definitions describe the format of the error log messages
/*      generated by MSCP and TMSCP devices.
/*-

/*
/*      Generic MSCP/TMSCP error log entry format
/*


aggregate GENERIC_MSCP_ERRLOG structure prefix MSLG$;
    CMD_REF longword unsigned;                  /* Command reference number
    UNIT word unsigned;                         /* Unit number
    SEQ_NUM word unsigned;                      /* Sequence Number
    FORMAT byte unsigned;                       /* Format
    FLAGS structure byte unsigned;              /* Error Log Message Flags
        LF_SQNRS bitfield mask;                 /*   Sequence Number Reset
        filler bitfield length 3 fill;
        LF_RPLER bitfield mask;                 /*   Error during replacement
        LF_BBR bitfield mask;                   /*   Bad block replacement request
        LF_CONT bitfield mask;                  /*   Operation continuing
        LF_SUCC bitfield mask;                  /*   Operation successful
    end FLAGS;
    EVENT word unsigned;                        /* Event Code
    constant (
          CNT_ERR                               {     Controller error
        , BUS_ADDR                              {     Host memory access error
        , DISK_TRN                              {     Disk transfer error (disks)
        , SDI                                   {     SDI error (disks)
        , SML_DSK                               {     Small disk error (disks)
        , TAPE_TRN                              {     Tape transfer error (tapes)
        , STI_ERR                               {     STI communication or command error (tapes)
        , STI_DEL                               {     STI driver error log (tapes)
        , STI_FEL                               {     STI formatter error log (tapes)
        , REPLACE                               {     Bad block replacement attempt (disks)
        ) equals 0 increment 1;
    CNT_ID quadword unsigned;                   /* Controller ID
    CNT_SVR byte unsigned;                      /* Controller software version
    CNT_HVR byte unsigned;                      /* Controller hardware version
    #cnt_err_base = .;
    MULT_UNT word unsigned;                     /* Multi-unit Code
    #bus_addr_base = .;
    UNIT_ID quadword unsigned;                  /* Unit ID
    UNIT_SVR byte unsigned;                     /* Unit software version
    UNIT_HVR byte unsigned;                     /* Unit hardware version
    #format_dependent = .;
    LEVEL byte unsigned;                        /* Level
    RETRY byte unsigned;                        /* Retry
    VOLSER_GAPCNT union fill;
        VOL_SER longword unsigned;              /* Volume Serial Number (disks)
        GAP_CNT longword unsigned;              /* Position - object count (tapes)
    end VOLSER_GAPCNT;
    #generic_disk_base = .;
    FMTR_SVR byte unsigned;                     /* Formatter software version
    FMTR_HVR byte unsigned;                     /* Formatter hardware version
    reserved word fill;
    #generic_tape_base = .;
end GENERIC_MSCP_ERRLOG;

/*
/*  Controller Error (MSLG$K_CNT_ERR)
/*

aggregate MSLG_CNT_ERR structure prefix MSLG$;

    filler byte dimension #cnt_err_base fill;
    CNT_ERR byte tag Z;                         /* Controller dependent data

end MSLG_CNT_ERR;

/*
/*  Host Memory Access Error (MSLG$K_BUS_ADDR)
/*

aggregate MSLG_BUS_ADDR structure prefix MSLG$;

    filler byte dimension #bus_addr_base fill;
    BUS_ADDR longword unsigned;                 /* Bus Address

end MSLG_BUS_ADDR;

/*
/*  Disk Transfer Error (MSLG$K_DISK_TRN)
/*

aggregate MSLG_DISK_TRN structure prefix MSLG$;

    filler byte dimension #generic_disk_base fill;
    HDR_CODE longword unsigned;                 /* Header Code
    DISK_TRN byte tag Z;                        /* Controller or disk dependent data

end MSLG_DISK_TRN;

/*
/*  SDI Error (MSLG$K_SDI)
/*

aggregate MSLG_SDI structure prefix MSLG$;

    filler byte dimension #generic_disk_base fill;
    hdr_code longword fill;                     /* Header Code (defined above)
    SDI byte unsigned dimension 12;             /* SDI Information

end MSLG_SDI;

/*
/*  Small Disk Error (MSLG$K_SML_DSK)
/*
```

```
SYSD

modu

/*+
/* M
/* T
/*-

aggr

end

/* T

aggr

end

end_
```

```
aggregate MSLG_SML_DSK structure prefix MSLG$;

    filler_1 byte dimension #format_dependent fill;
    SDE_CYL word unsigned;                      /* Cylinder
    filler_2 byte dimension #generic_disk_base-. fill;
    SML_DSK byte tag Z;                         /* Controller or device dependent

end MSLG_SML_DSK;

/*
/*  Tape Transfer Error (MSLG$K_STI_ERR)
/*
/*      There are no special field definitions for tape transfer errors at this time.

/*
/*  STI communication or command failure (MSLG$K_STI_ERR)
/*  STI drive error log (MSLG$K_STI_DEL)
/*  STI formatter error log (MSLG$K_STI_FEL)
/*

aggregate MSLG_STI_ERR structure prefix MSLG$;

    filler byte dimension #generic_tape_base fill;
    STI byte unsigned dimension 20;             /* STI Information

end MSLG_STI_ERR;

/*
/*  Bad Block Replacement Attempted (MSLG$K_REPLACE)
/*

aggregate MSLG_REPLACE structure prefix MSLG$;

    filler_1 byte dimension #format_dependent fill;
    RPL_FLGS structure word unsigned;           /* Replace Flags
        bit_fill bitfield length 10 fill;
        LFR_BR bitfield mask;                   /*  Bad RBN
        LFR_RI bitfield mask;                   /*  RCT inconsistent
        LFR_RF bitfield mask;                   /*  Reformat error
        LFR_TE bitfield mask;                   /*  Tertiary revector
        LFR_FE bitfield mask;                   /*  Forced error (data not recovered)
        LFR_RP bitfield mask;                   /*  Replace attempted (block really bad)
    end RPL_FLGS;
    filler_2 byte dimension #generic_disk_base-. fill;
    BAD_LBN longword unsigned;                  /* Bad LBN
    OLD_RBN longword unsigned;                  /* Previous RBN
    NEW_RBN longword unsigned;                  /* New RBN
    CAUSE word unsigned;                        /* Event code causing replacement

end MSLG_REPLACE;

end_module $MSLGDEF;
```

```
module $MSCPDEF;

/*++
/* MSCP (Mass Storage Control Protocol) Definitions
/*
/*      These definitions describe the format of the command and end message
/*      packets exchanged under MSCP between the host and the controller.
/*--

aggregate GENERIC_MSCP structure prefix MSCP$;

    CMD_REF longword unsigned;                      /* Command reference number
    UNIT structure word unsigned;                   /* Unit number
        EU_NO_OVERLAY union fill;
            EU_NO bitfield length 8 mask;           /* Emulated unit number
            EU_SUB_NO structure fill;
                EU_SUBU bitfield length 3 mask;     /* Old-style unit number
                EU_SUBC bitfield length 5 mask;     /* Old-style controller subtype
                constant (                          {  subtype values:
                    EMS_CNSL,                       {       Console
                    EMS_RP,                         {       RP04/05/06
                    EMS_RM,                         {       RM03/05/80/RP07
                    EMS_RK,                         {       RK06/07
                    EMS_RL,                         {       RL01/02
                    EMS_RX,                         {       RX211
                    EMS_FD1,                        {       Foreign disk type 1
                    EMS_FD2,                        {       Foreign disk type 2
                    EMS_FD3,                        {       Foreign disk type 3
                    EMS_FD4,                        {       Foreign disk type 4
                    EMS_FD5,                        {       Foreign disk type 5
                    EMS_FD6,                        {       Foreign disk type 6
                    EMS_FD7,                        {       Foreign disk type 7
                    EMS_FD8                         {       Foreign disk type 8
                    ) equals 0 increment 1;
            end EU_SUB_NO;
        end EU_NO_OVERLAY;
        EU_CTYPE bitfield length 4 mask;            /* Emulated controller type
        constant (                                  {  controller type values:
            EMD_OLD,                                {       old-style (highest unit number is 7)
            EMD_UDA,                                {       UDA
            EMD_HSC,                                {       HSC
            EMD_AZT,                                {       RC25 (AZTEC)
            EMD_RDRX,                               {       RD/RX
            EMD_EMUL                                {       Emulated
            ) equals 0 increment 1;
        EU_DESIG bitfield length 3 mask;            /* Emulated controller designator
        SHADOW bitfield mask;                       /* Shadow unit
    end UNIT;
    reserved word fill;
    OPCODE structure byte unsigned;                 /* MSCP operation code
        code bitfield length 3 fill;                {  function code
        type bitfield length 3 fill;                {  immediate / sequential / non-sequential
        OP_ATTN bitfield mask;                      /* Attention message
        OP_END bitfield mask;                       /* End message
    end OPCODE;
    MODIFIERS_STATUS union fill;
```

```
        MODIFIERS structure fill;
            reserved byte fill;
            #modifier_base =.;                          {  base for modifiers setup
            MODIFIER word unsigned;                     /* MSCP command modifiers
        end MODIFIERS;
        FLAGS_STATUS structure fill;
            FLAGS structure byte unsigned;              /* End message flags
                filler bitfield length 2 fill;
                EF_PLS bitfield mask;                   /* Position Lost (tapes only)
                EF_EOT bitfield mask;                   /* End of Tape Encountered (tapes only)
                EF_SEREX bitfield mask;                 /* Serious exception (tapes only)
                EF_ERLOG bitfield mask;                 /* Error log generated
                EF_BBLKU bitfield mask;                 /* Bad block unreported (disks only)
                EF_BBLKR bitfield mask;                 /* Bad block reported (disks only)
            end FLAGS;
            #status_base =.;                            {  base for status setup
            STATUS structure word unsigned;             /* End message status
                ST_MASK bitfield length 5 mask;         /* Status code bits
                constant (                              {  status code values:
                        ST_SUCC,                        {       Success
                        ST_ICMD,                        {       Invalid command
                        ST_ABRTD,                       {       Command aborted
                        ST_OFFLN,                       {       Unit-offline
                        ST_AVLBL,                       {       Unit-available
                        ST_MFMTE,                       {       Media format error
                        ST_WRTPR,                       {       Write protected
                        ST_COMP,                        {       Compare error
                        ST_DATA,                        {       Data error
                        ST_HSTBF,                       {       Host buffer access error
                        ST_CNTLR,                       {       Controller error
                        ST_DRIVE,                       {       Drive error
                        ST_FMTER,                       {       Formatter error (tapes only)
                        ST_BOT,                         {       BOT encountered (tapes only)
                        ST_TAPEM                        {       Tape mark encountered (tapes only)
                        ) equals 0 increment 1,
                        ST_SHST equals 12, (            {       Shadow set state change (disks only)
                        ST_RDTRN,                       {       Record data truncated (tapes only)
                        ST_PLOST,                       {       Position lost (tapes only)
                        ST_PRESE,                       {       Previous serious exception (tapes only)
                        ST_LED,                         {       LEOT detected (tapes only)
                        ST_BBR                          {       Bad block replacement completed (disks only)
                        ) equals 16 increment 1, (
                        ST_DIAG,                        {       Message from internal diagnostic
                        ST_SBCOD                        {       Subcode multiplier
                        ) equals 31 increment 1;
                ST_SBCOD bitfield length 11 mask;       /* Subcode bits
                                                        {  Subcode values defined seperately below

            end STATUS;
        end FLAGS_STATUS;
    end MODIFIERS_STATUS;

    #end_basic_packet = .;

    /* MSCP Command Operation Codes (defined in alphabetical order)

    constant OP_ABORT equals 1;                         /* Abort
    constant OP_ACCES equals 16;                        /* Access
    constant OP_AVAIL equals 8;                         /* Available
    constant OP_CMPCD equals 17;                        /* Compare Controller Data
    constant OP_COMP  equals 32;                        /* Compare Host Data
    constant OP_DTACP  equals 11;                       /* Determine Access Paths
    constant OP_ERASE equals 18;                        /* Erase
    constant OP_ERGAP equals 22;                        /* Erase Gap (tapes only)
    constant OP_FLUSH equals 19;                        /* Flush
    constant OP_GTCMD equals 2;                         /* Get Command Status
    constant OP_GTUNT equals 3;                         /* Get Unit Status
    constant OP_ONLIN equals 9;                         /* Online
    constant OP_READ  equals 33;                        /* Read
    constant OP_REPLC equals 20;                        /* Replace
    constant OP_REPOS equals 37;                        /* Reposition (tapes only)
    constant OP_STCON equals 4;                         /* Set Controller Characteristics
    constant OP_STUNT equals 10;                        /* Set Unit Characteristics
    constant OP_WRITE equals 34;                        /* Write
    constant OP_WRITM equals 36;                        /* Write Tape Mark

    /* MSCP End Message Codes

    constant OP_END   equals %x80;                      /* End Message Flag
    constant OP_SEREX equals 7;                         /* Serious Execption (end message only)

    /* MSCP Attention Message Codes (listed in alphabetical order)

    constant OP_ACPTH equals 66;                        /* Access Path
    constant OP_AVATN equals 64;                        /* Available
    constant OP_DUPUN equals 65;                        /* Duplicate Unit Number
    constant OP_RWATN equals 67;                        /* Rewind (tapes only)

end GENERIC_MSCP;

aggregate MSCP_MODIFIERS structure prefix MSCP$;

  filler byte dimension #modifier_base fill;
  ALL_MODIFIERS union fill;

    /* Generic MSCP Modifiers

    GENERIC_MODIFIERS structure fill;                   {  Generic command modifiers:
        filler bitfield length 8 fill;
        MD_SEREC bitfield mask;                         /*   Suppress error recovery
        MD_SECOR bitfield mask;                         /*   Suppress error correction
        filler bitfield length 3 fill;
        MD_CLSEX bitfield mask;                         /*   Clear serious exception
        MD_COMP bitfield mask;                          /*   Compare
    end GENERIC_MODIFIERS;

    DISK_MODIFIERS structure fill;                      {  Generic disk command modifiers:
        filler bitfield length 4 fill;
        MD_WRSEQ bitfield mask;                         /*   Write shadow set 1 unit at a time
        MD_WBKVL bitfield mask;                         /*   Write-back (volatile)
        MD_WBKNV bitfield mask;                         /*   Write-back (non-volatile)
        MD_SSHDW bitfield mask;                         /*   Suppress Shadowing
        filler bitfield length 2 fill;
        MD_SCCHL bitfield mask;                         /*   Suppress caching (low speed)
        MD_SCCHH bitfield mask;                         /*   Suppress caching (high speed)
        MD_ERROR bitfield mask;                         /*   Force error
        filler bitfield length 2 fill;
        MD_EXPRS bitfield mask;                         /*   Express request
    end DISK_MODIFIERS;

    TAPE_MODIFIERS structure fill;                      { Generic tape command modifiers:
        filler bitfield length 1 fill;
        MD_REWND bitfield mask;                         /*   Rewind
        MD_OBJCT bitfield mask;                         /*   Object count
        MD_REVRS bitfield mask;                         /*   Reverse
        MD_UNLOD bitfield mask;                         /*   Unload
        MD_EXCLU bitfield mask;                         /*   Exclusive
        MD_IMMED bitfield mask;                         /*   Request immediate completion
        MD_DLEOT bitfield mask;                         /*   Request detect LEOT
    end TAPE_MODIFIERS;

    AVAIL_MODIFIERS structure fill;                     { Available command modifiers:
        MD_ALLCD bitfield mask;                         /*   All class drivers
        MD_SPNDW bitfield mask;                         /*   Spin down
        MD_DSOLV bitfield mask;                         /*   Disolve shadow set
    end AVAIL_MODIFIERS;

    FLUSH_MODIFIERS structure fill;                     { Flush command modifiers:
        MD_FLENU bitfield mask;                         /*   Flush entire unit
        MD_VOLTL bitfield mask;                         /*   Flush volitile only
    end FLUSH_MODIFIERS;

    GTUNT_MODIFIERS structure fill;                     { Get unit status command modifiers:
        MD_NXUNT bitfield mask;                         /*   Next unit
    end GTUNT_MODIFIERS;

    ONLIN_STUNT_MODIFIERS structure fill;               { Online and set unit characteristics modifiers:
        MD_RIP bitfield mask;                           /*   Allow self-destruct (online only)
        MD_IGNMF bitfield mask;                         /*   Ignore media format error (online only)
        MD_STWRP bitfield mask;                         /*   Enable Set Write Protect
        MD_CLWBL bitfield mask;                         /*   Clear Write-Back Data Lost
        MD_SHDSP bitfield mask;                         /*   Shadow Unit Specified
    end ONLIN_STUNT_MODIFIERS;

    REPLC_MODIFIERS structure fill;                     { Replace command modifiers:
        MD_PRIMR bitfield mask;                         /*   Primary replacement block
    end REPLC_MODIFIERS;

  end ALL_MODIFIERS;

end MSCP_MODIFIERS;

aggregate MSCP_SUBCODES structure prefix MSCP$;

  filler byte dimension #status_base fill;
  ALL_SUBCS union fill;

{ NOTE:
{       Many of the subcode values are defined such that they produce bit
{       fields.  This is not a requirement in the MSCP specification.  So long
{       as new subcodes continue to produce bit fields, the bit field
{       definitions here may remain.  When, as, and if, bit fields are no
{       longer produced, the bit field definitions MUST be removed here and
{       the code which breaks must be fixed.

    /* Success Subcode Values

    SC_SUCC structure fill;
        constant SC_NORML equals 0;                         /* Normal
        constant SC_SDIGN equals 1;                         /* Spin Down IGNored
        constant SC_STCON equals 2;                         /* STill CONnected
        constant SC_DUPUN equals 4;                         /* DUPlicate UNit number
        constant SC_ALONL equals 8;                         /* ALready ONLine
        constant SC_STONL equals 16;                        /* STill ONLine
        constant SC_EOT   equals 32;                        /* EOT encountered (tapes only)
        constant SC_INREP equals 32;                        /* INcomplete REPlacement (disks only)
        constant SC_IVRCT equals 64;                        /* InValid RCT (disks only)
        bit_fields union fill;
            fields_1 structure fill;
                filler bitfield length 5 fill;
                SC_SDIGN bitfield mask;                     /* Spin Down IGNored
                SC_STCON bitfield mask;                     /* STill CONnected
                SC_DUPUN bitfield mask;                     /* DUPlicate UNit number
                SC_ALONL bitfield mask;                     /* ALready ONLine
                SC_STONL bitfield mask;                     /* STill ONLine
                SC_EOT bitfield mask;                       /* EOT encountered (tapes only)
            end fields_1;
            field_2 structure fill;
                filler bitfield length 10 fill;
                SC_INREP bitfield mask;                     /* INcomplete REPlacement (disks only)
                SC_IVRCT bitfield mask;                     /* InValid RCT (disks only)
            end fields_2;
        end bit_fields;
    end SC_SUCC;

    /* Invalid Command Subcode Values

    constant SC_INVML equals 0;                             /* INValid Message Length

    /* Unit-Offline Subcode Values

    SC_OFFLN structure fill;
        constant SC_UNKNO equals 0;                         /* UNKoNOwn unit or online to another controller
        constant SC_NOVOL equals 1;                         /* NO VOLume mounted or drive disabled (RUN/STOP)
        constant SC_INOPR equals 2;                         /* unit is INOPeRative
                                                            {  duplicate unit number (already defined above)
        constant SC_UDSBL equals 8;                         /* Unit disabled by field service or diagnostic
        filler bitfield length 5 fill;
        SC_NOVOL bitfield mask;                             /* NO VOLume mounted or drive disabled (RUN/STOP)
        SC_INOPR bitfield mask;                             /* unit is INOPeRative
        dupun bitfield fill;                                {  duplicate unit number (already defined above)
        SC_UDSBL bitfield mask;                             /* Unit disabled by field service or diagnostic
    end SC_OFFLN;

/* Unit-Available Subcode Values
```

```
SYSD

cons
cons
cons

end_
```

```
    constant SC_NOMEMB equals 1;                           { No members

    /* Write-Protected Subcode Values

    SC_WRTPR structure fill;
        constant SC_DATAL equals 8;                        /* Unit is DATA Loss write protected
        constant SC_SOFTW equals 128;                      /* Unit is SOFTWare protected
        constant SC_HARDW equals 256;                      /* Unit is HARDWare protected
        filler bitfield length 8 fill;
        SC_DATAL bitfield mask;                            /* Unit is DATA Loss write protected
        filler bitfield length 3 fill;
        SC_SOFTW bitfield mask;                            /* Unit is SOFTWare protected
        SC_HARDW bitfield mask;                            /* Unit is HARDWare protected
    end SC_WRTPR;

    /* Data Error Subcode Values

    constant SC_FRCER equals 0;                            /* FoRDed ERror

    /* Host Buffer Access Error Subcode Values

    constant (
        SC_ODDTA,                                          { Odd transfer address
        SC_ODDBC,                                          { Odd BCNT
        SC_NXM,                                            { Non-existant memory
        SC_MPAR,                                           { Host memory parity
        SC_IVPTE,                                          { Invalid page table entry
        SC_IVBFN,                                          { Invalid buffer name
        SC_BLENV,                                          { Buffer length violation
        SC_ACVIO                                           { Access control violation
        ) equals 1 increment 1;

    /* Controller Error Subcode Values

    constant (
        SC_DLATE,                                          { Date late
        SC_EDCER,                                          { EDC error
        SC_DTSTR,                                          { Data structure error
        SC_IEDC,                                           { Internal EDC error
        SC_LACIN,                                          { LESI adapter card input parity
        SC_LACOU,                                          { LESI adapter card output parity
        SC_LACCB,                                          { LESI adapter card "cable in place" not asserted
        SC_OVRUN,                                          { Controller overrun or underrun
        SC_MEMER,                                          { Controller memory error
        SC_REMRSRC                                         { Insufficient resources
        ) equals 1 increment 1;

    /* Bad Block Replacement Subcode Values

    constant (
        SC_BBROK,                                          { Bad block replacement successful
        SC_NOTRP,                                          { Block tested ok, not replaced
        SC_RPLFL,                                          { REPLACE command failure
        SC_ICRCT,                                          { Inconsistant RCT
        SC_DRVER                                        {  Drive error
        ) equals 0 increment 1;

  end ALL_SUBCS;

end MSCP_SUBCODES;

/* Definitions for MSCP Transfer Commands

aggregate TRANSFER_COMMANDS structure prefix MSCP$;
    base byte dimension #end_basic_packet fill;

    BYTE_CNT longword unsigned;                         /* Byte count
    BUFFER byte unsigned dimension 12;                  /* Buffer descriptor
    DISK_TAPE union fill;
        DISK structure fill;
            LBN structure longword unsigned;            /* Logical block number
                FRST_BAD longword unsigned;             /*   First bad block
            end LBN;
        end DISK;
        TAPE structure fill;
            POSITION longword unsigned;                 /* Position (object count)
            TAPEREC longword unsigned;                  /* Tape record byte count
        end TAPE;
    end DISK_TAPE;

end TRANSFER_COMMANDS;

/* Definitions for Abort and Get Command Status Commands and End Messages

aggregate ABORT_GTCMD structure prefix MSCP$;
    base byte dimension #end_basic_packet fill;
    OUT_REF longword unsigned;                          /* Outstanding reference number
    CMD_STS longword unsigned;                          /* Command status
end ABORT_GTCMD;

/* Definitions for the Get Unit Status Command and End Message

aggregate GTUNT structure prefix MSCP$;

    base byte dimension #end_basic_packet fill;
    MULT_UNT word unsigned;                             /* Multi-unit code
    UNT_FLGS structure word unsigned;                   /* Unit flags
        UF_CMPRD bitfield mask;                         /*   Compare reads
        UF_CMPWR bitfield mask;                         /*   Compare writes
        UF_576 bitfield mask;                           /*   576 byte sectors [disks only]
        filler bitfield fill;
        UF_VARSP bitfield mask;                         /*   Variable speed unit [tapes only]
        UF_VSMSU bitfield mask;                         /*   Variable speed mode suppression [tapes only]
        UF_WBKNV bitfield mask;                         /*   Write-back (non-volatile) [disks only]
        UF_RMVBL bitfield mask;                         /*   Removeable media [disks only]
        UF_WRTPD bitfield mask;                         /*   Write protect (data loss)
        UF_SSMST bitfield mask;                         /*   Shadow set master
        UF_SCCHL bitfield mask;                         /*   Suppress caching (Low speed) [disks only]
        UF_SCCHH bitfield mask;                         /*   Suppress caching (High speed) [disks only]
        UF_WRTPS bitfield mask;                         /*   Write protect (software)
        UF_WRTPH bitfield mask;                        /*   Write protect (hardware)
        UF_SSMEM bitfield mask;                        /*   Shadow set member
        UF_REPLC bitfield mask;                        /*   Controller initiated bad block replacement [disks only]
    end UNT_FLGS;
    reserved longword fill;
    UNIT_ID structure quadword unsigned;               /* Unit identifier
        EXCL_LBA longword unsigned;                    /* Excluded LBN area address [disks only]
        EXCL_LBC word unsigned;                        /* Excluded LBN block count [disks only]
    end UNIT_ID;
    DEV_PARM_OVERLAY union fill;
        DEV_PARM longword unsigned;                    /* Device dependent parameters
        MEDIA_ID structure longword unsigned;          /* Media type identifier
            MTYP_N bitfield length 7 mask;             /*   Media # (i.e. 7 of RK07)
            MTYP_A2 bitfield length 5 mask;            /*   Media name char.
            MTYP_A1 bitfield length 5 mask;            /*   Media name continued
            MTYP_A0 bitfield length 5 mask;            /*        "     "      "
            MTYP_D1 bitfield length 5 mask;            /*   Dev mnemonic char.
            MTYP_D0 bitfield length 5 mask;            /*   Mnemonic continued
        end MEDIA_ID;
    end DEV_PARM_OVERLAY;
    DISK_TAPE_CMD union fill;
        DISK_CMD structure fill;
            SHDW_UNT word unsigned;                    /* Shadow unit
            SPD_STS union fill;
                COPY_SPD word unsigned;                /* Copy speed
                constant (                             {  Copy speeds:
                    CS_NOCPY,                          {       no copy
                    CS_COPY,                           {       regular copy
                    CS_MGCPY                           {       merge copy
                    ) equals 0 increment 1;
                SHDW_STS word unsigned;                /* Shadow unit status
                    COPYIP bitfield mask;              /*  Shadow copy in progress
            end SPD_STS;
        end DISK_CMD;
        TAPE_CMD structure fill;
            FORMAT structure word unsigned;            /* Format
                TF_800 bitfield mask;                  /*  NRZI 800 bpi
                TF_PE bitfield mask;                   /*  Phase encoded 1600 bpi
                TF_GCR bitfield mask;                  /*  Group code recording 6250 bpi
            end FORMAT;
            SPEED word unsigned;                       /* Speed
        end TAPE_CMD;
    end DISK_TAPE_CMD;
    #onlin_stunt_base = .; {-- marker for beginning of online & set unit characteristics defs.

    {{{{ The longest command ends here. }}}}
    #longest_command = .;

    DISK_TAPE_END union fill;
        DISK_END structure fill;
            TRACK word unsigned;                       /* Track size
            GROUP word unsigned;                       /* Group size
            CYLINDER word unsigned;                    /* Cylinder size
            UNIT_SVR byte unsigned;                    /* Unit software version
            UNIT_HVR byte unsigned;                    /* Unit hardware version
            RCT_SIZE word unsigned;                    /* RCT size
```

```
SYSD

/* F

aggr

end

/* V

aggr

end

/* \
            RBNS byte unsigned;                             /* RBNs per track
            RCT_CPYS byte unsigned;                         /* Number of RCT copies
        end DISK_END;
        TAPE_END structure fill;
            FORMENU word unsigned;                          /*  Format menu
        end TAPE_END;
    end DISK_TAPE_END;

    {{{{ The longest end-message ends here. }}}}
    #longest_end_message = .;

end GTUNT;

/* Definitions for Online and Set Unit Characteristics Command and End Messages

aggregate ONLIN_STUNT structure prefix MSCP$;

    marker byte dimension #onlin_stunt_base fill;
    DISK_TAPE union fill;
        DISK structure fill;
            UNT_SIZE longword unsigned;                     /* Unit size
            VOL_SER longword unsigned;                      /* Volume serial number
        end DISK;
        TAPE structure fill;
            MAXWTREC longword unsigned;                     /* Maximum write record size
            NOISEREC word unsigned;                         /* Noise record
        end TAPE;
    end DISK_TAPE;

end ONLIN_STUNT;

/* Definitions for the Replace Command and End Message (disks only)

aggregate REPLC structure prefix MSCP$;
    base byte dimension #end_basic_packet fill;
    RBN longword unsigned;                                  /* Replacement block number
end REPLC;

/* Definitions for the Reposition Command and End Message (tapes only)

aggregate REPOS structure prefix MSCP$;
    base byte dimension #end_basic_packet fill;
    CMDEND union fill;
        CMD structure fill;
            REC_CNT longword unsigned;                      /* Record/Object count
            TMGP_CNT longword unsigned;                     /* Tape mark count
        end CMD;
        ENDMSG structure fill;
            RCSKIPED longword unsigned;                     /* Records skipped
            TMSKIPED longword unsigned;                     /* Tape markes skipped
        end ENDMSG;
    end CMDEND;
end REPOS;

/* Definitions for the Set Controller Characteristics Command and End Message
```

```
aggregate STCON structure prefix MSCP$;

    filler byte dimension 4 fill;
    CNT_ALCS byte unsigned;                         /* Allocation class
    filler byte dimension #end_basic_packet-5 fill;
    VERSION word unsigned;                          /* MSCP version
    CNT_FLGS structure word unsigned;               /* Controller flags
        CF_576 bitfield mask;                       /*   576 byte sectors [disks only]
        CF_SHADW bitfield mask;                     /*   Shadowing [disks only]
        CF_MLTHS bitfield mask;                     /*   Multi-Host
        filler bitfield length 1 fill;
        CF_THIS bitfield mask;                      /*   Enable this host's error log
        CF_OTHER bitfield mask;                     /*   Enable other host's error log
        CF_MISC bitfield mask;                      /*   Enable miscellaneous error log
        CF_ATTN bitfield mask;                      /*   Enable attention messages
        filler bitfield length 7 fill;
        CF_REPLC bitfield mask;                     /*   Controller Initiated Bad Block Replacement [disks only]
    end CNT_FLGS;
    HST_TMO structure word unsigned;                /* Host timeout
        CNT_TMO word unsigned;                      /* Controller timeout
    end HST_TMO;
    CNT_SVR byte unsigned;                          /* Controller software version
    CNT_HVR byte unsigned;                          /* Controller hardware version
    TIME structure quadword unsigned;               /* Quad-word date-time
        CNT_ID quadword unsigned;                   /* Controller ID
    end TIME;

    /*  Controller and Unit identifier Classes. (Device Class)
    constant CL_CNTRL equals 1;                     /* MSCP Controller
    constant CL_DISK  equals 2;                     /* Disk Class Device
    constant CL_TAPE  equals 3;                     /* Tape Class Device
    constant CL_D144  equals 4;                     /* DEC144 Disk Class Device

    /*  MSCP Controller Model
    constant CM_HSC50 equals 1;                     /* HSC50
    constant CM_UDA50 equals 2;                     /* UDA50
    constant CM_RC25  equals 3;                     /* RC25 (AZTEC)
    constant CM_EMULA equals 4;                     /* Emulator
    constant CM_TU81  equals 5;                     /* TU81
    constant CM_UDA52 equals 6;                     /* UDA52 (UDA50A old name)
    constant CM_UDA50A equals 6;                    /* UDA50A
    constant CM_RDRX  equals 7;                     /* RD/RX
    constant CM_TOPS  equals 8;                     /* TOPS 10/20 Emulator
    constant CM_TK50  equals 9;                     /* TK50
    constant CM_RUX50 equals 10;                    /* RUX50
    constant CM_RC26  equals 11;                    /* RC26
    constant CM_AIO   equals 12;                    /* AURORA I/O
    constant CM_QDA50 equals 13;                    /* QDA50
    constant CM_BDA   equals 14;                    /* BDA
    constant CM_BSA   equals 15;                    /* BSA
    constant CM_CDR50 equals 16;                    /* CDR50
    constant CM_QDA25 equals 17;                    /* QDA25

    constant MXCMDLEN equals #longest_command;      /* Longest Command
    constant MXCMDLEN equals #longest_command tag C;    /* Longest Command
    constant LEN equals #longest_end_message;         /* Longest End Message
    constant LEN equals #longest_end_message tag C;   /* Longest End Message

end STCON;

end_module $MSCPDEF;
```

SYSD

```
module $MVLDEF;

/*+
/* MAGNETIC TAPE VOLUME LIST
/* THIS STRUCTURE DESCRIBES THE VOLUMES IN A VOLUME SET
/*-

aggregate MVLDEF structure prefix MVL$;
    VCB longword unsigned;                              /*ADDRESS OF VCB
    FILL_1 longword fill prefix MVLDEF tag $$;          /*SPARE
    SIZE word unsigned;                                 /*SIZE OF STRUCTURE
    TYPE byte unsigned;                                 /*TYPE OF STRUCTURE
    NVOLS byte unsigned;                                /*NUMBER OF VOLUMES IN VOLUME SET
    SET_ID character length 6;                          /*FILE SET ID FOR THE VOLUME SET
    VOL_ACC byte unsigned;                              /*VOLUME ACCESSIBILTY CHARACTER DEFAULT
    MOU_PRV_OVERLAY union fill;
        MOU_PRV byte unsigned;                          /*USER'S MOUNT TIME PRIVILEGES
        MOU_PRV_BITS structure fill;
            VOLPRO bitfield;                            /*VOLPRO PRIVILEGE
            OVRPRO bitfield;                            /*OVERRIDE PRIVILEGE (BYPASS,SYSPRV,VOLPRO)
            OPER bitfield;                              /*OPER PRIVILEDGE
        end MOU_PRV_BITS;

    end MOU_PRV_OVERLAY;
        VOLOWNER character length 14;                   /* VOL1 ONER IDENTIFIER  FIELD
        STDVER byte unsigned;                           /* ANSI VERSION OF VOLUME SET
        FILL_2 byte fill prefix MVLDEF tag $$;          /* SPARE
        constant FIXLEN equals . prefix MVL$ tag K;     /*LENGTH OF FIXED AREA OF STRUCTURE
        constant FIXLEN equals . prefix MVL$ tag C;     /*LENGTH OF FIXED AREA OF STRUCTURE
end MVLDEF;

/* THE FOLLOWING STRUCTURE IN REPEATED IN MVL FOR EACH REEL IN VOLUME SET

aggregate MVLDEF1 structure prefix MVL$;
    VOLLBL character length 6;                          /*VOLUME LABEL
    RVN byte unsigned;                                  /*RELATIVE UNIT NUMBER
    STATUS_OVERLAY union fill;
        STATUS byte unsigned;                           /*STATUS OF VOLUME
        constant "LENGTH" equals . prefix MVL$ tag K;   /*LENGTH OF STRUCTURE
        constant "LENGTH" equals . prefix MVL$ tag C;   /*LENGTH OF STRUCTURE
        STATUS_BITS structure fill;
            MOUNTED bitfield;                           /*REEL IS MOUNTED
            UNUSED bitfield;                            /*IS THIS ENTRY IN USE
            OVERIDE bitfield;                           /*CAN OVERRIDE PROTECTION ON THIS REEL
        end STATUS_BITS;
    end STATUS_OVERLAY;
end MVLDEF1;

end_module $MVLDEF;
```

```
SYSD

modu

/*+
/* S
/*-

aggr





end

end_
```

```
module $MVMSLDEF;

/*++
/*  $MVMSLDEF - mount verification messages list structure definition
/*
/*      The MVMSL provides a mechanism for communicating information about
/*      mount verification messages to device driver special mount
/*      verification processing routines.
/*--
aggregate MVMSLDEF structure prefix MVMSL$ origin MSG_CODE;

    MAXIDX byte unsigned;                       /* Maximum legal MVMSL index.
    SNDMSGOFF longword;                         /* Offset from MVMSL base to SEND_MESSAGE routine
    MSG_CODE word unsigned;                     /* The MSG$_ code for this entry.
    FLAGS structure word unsigned;              /* Processing flags:
        NOSUFFIX bitfield mask;                 /*      Do not add suffix.
        SUPRESS bitfield mask;                  /*      May be suppressed.
    end FLAGS;
    TEXTOFF longword;                           /* Offset from MVMSL base to ASCIC message text.
    constant 'LENGTH' equals .;                 /* Length of a MVMSL entry.

end MVMSLDEF;

end_module $MVMSLDEF;
```

```
module $NBIADEF;

aggregate NBIADEF structure prefix NBIA$;

/**
/*      Nautilus NBIA register definitions
/*
/* The NBIA sits in an NMI nexus and can connect one or two BIs to a Nautilus.
/**

    CSR0_OVERLAY union fill;
        CSR0 longword unsigned;                 /* Control and Status
        CSR0_FIELD_OVERLAY union fill;
            NAC byte unsigned;                  /* Adapter Type Field
            CSR0_BITS structure fill;
                FILL_0 bitfield length 8 fill prefix NBIADEF tag $$;
                BIOPU bitfield mask;            /* BI0 Power Up
                NBIVO bitfield length 6;        /* Vector Offset Register
                NPE bitfield mask;              /* NBI Parity Error
                BILP bitfield mask;             /* BIIC Loopback
                FNPE bitfield mask;             /* Force NBI Parity error
                FDB  bitfield mask;             /* Force DMA busy
                FLIP_29_22 bitfield mask;       /* Maintenance Magic
                FILL_1 bitfield length 1 fill prefix NBIADEF tag $$;
                NIE bitfield mask;              /* NBI Interrupt Enable
                FILL_2 bitfield length 2 fill prefix NBIADEF tag $$;
                TOI  bitfield length 3;         /* Time-out Interrupt
                TDF  bitfield mask;             /* Transmitter During Fault
                WDSF bitfield mask;             /* Write Data Sequence Fault
                RDSF bitfield mask;             /* Read Data Sequence Fault
                CPF  bitfield mask;             /* Control Parity Fault
                DPF  bitfield mask;             /* Data Parity Fault
            end CSR0_BITS;
        end CSR0_FIELD_OVERLAY;
    end CSR0_OVERLAY;

    CSR1_OVERLAY union fill;
        CSR1 longword unsigned;                 /* NBIA CSR1
        CSR1_BITS structure fill;
            ADIN bitfield mask;                 /* Adaptor Init
            BI0P bitfield mask;                 /* BI0 Present
            BI1P bitfield mask;                 /* BI1 Present
            BI0_PE bitfield mask;               /* BI0 Parity Error
            BI1_PE bitfield mask;               /* BI1 Parity Error
            FILL_3 bitfield length 3 fill prefix NBIADEF tag $$;
            BI1PU bitfield mask;                /* BI1 Power Up
            NAWR  bitfield mask;                /* NBIA Wraparound (Maint Magic)
            FBPE  bitfield mask;                /* Force NBIB Parity Error
            FILL_4 bitfield length 21 fill prefix NBIADEF tag $$;
        end CSR1_BITS;
    end CSR1_OVERLAY;

    BI0I longword unsigned;                     /* BI0 Stop Register

    BI1I longword unsigned;                     /* BI1 Stop Register
```

SYSD

end_

```
    BR4VR longword unsigned;                    /* BR4 Vector Register
    BR5VR longword unsigned;                    /* BR5 Vector Register
    BR6VR longword unsigned;                    /* BR6 Vector Register
    BR7VR longword unsigned;                    /* BR7 Vector Register
end NBIADEF;
end_module $NBIADEF;
```

```
module $NAFDEF;

/*++
/*
/* Structure for network proxy login file, NETUAF.DAT
/*
/*--


aggregate NAFDEF structure prefix NAF$;
    REMNAME structure character length 64;      /* Combined nodename and remote username
        NODE character length 32;               /* Node name
        REMUSER character length 32;            /* Remote username
    end REMNAME;
    LOCALUSER character length 32;              /* Local username
    FLAGS structure longword;                   /* Flags longword
        TASK bitfield mask;                     /* Allow task=0 access
        BATCH bitfield mask;                    /* Allow batch jobs
        INTERACTIVE bitfield mask;              /* Allow interactive login
    end FLAGS;
    constant "LENGTH" equals . tag K;           /* Length of record
    constant "LENGTH" equals . tag C;           /* Length of record
end NAFDEF;

end_module $NAFDEF;
```

SYSD

end

end_

```
module $NDTDEF;

/*+
/* NEXUS DEVICE AND ADAPTER TYPE CODES
/*-


                                                        /*DEFINE CONSTANT TYPE CODES
constant MEM4NI equals 8  prefix NDT tag $;             /*MEMORY, 4K NOT INTERLEAVED
constant MEM4I  equals 9  prefix NDT tag $;             /*MEMORY, 4K INTERLEAVED
constant MEM16NI        equals +%X10  prefix NDT tag $; /*MEMORY, 16K NOT INTERLEAVED
constant MEM16I equals +%X11  prefix NDT tag $;         /*MEMORY, 16K INTERLEAVED
constant MEM1664NI      equals +%X12  prefix NDT tag $; /*MEMORY, 16K AND 64K MIXED
constant MB     equals +%X20  prefix NDT tag $;         /*MBA 0,1,2, OR 3
constant UB0    equals +%X28  prefix NDT tag $;         /*UB ADAPTER OR INTERCONNECT 0,
constant UB1    equals +%X29  prefix NDT tag $;         /* 1,
constant UB2    equals +%X2A  prefix NDT tag $;         /* 2,
constant UB3    equals +%X2B  prefix NDT tag $;         /* OR 3
constant CI     equals +%X38  prefix NDT tag $;         /*CI780'S, CI750'S
constant MPM0   equals +%X40  prefix NDT tag $;         /*MULTIPORT MEMORY 0,
constant MPM1   equals +%X41  prefix NDT tag $;         /* 1,
constant MPM2   equals +%X42  prefix NDT tag $;         /* 2,
constant MPM3   equals +%X43  prefix NDT tag $;         /* OR 3
constant DR32   equals +%X30  prefix NDT tag $;         /*DR32 0,1,2,...
constant MEM64NIL       equals +%X68  prefix NDT tag $; /*64K NON-INTERLEAVED MEM, LOWER CONTROLLER
constant MEM64EIL       equals +%X69  prefix NDT tag $; /*64K EXTERNALLY INTERLEAVED MEM, LOWER
constant MEM64NIU       equals +%X6A  prefix NDT tag $; /*64K NON-INTERLEAVED MEM, UPPER CONTROLLER
constant MEM64EIU       equals +%X6B  prefix NDT tag $; /*64K EXTERNALLY INTERLEAVED MEM, UPPER
constant MEM64I equals +%X6C  prefix NDT tag $;         /*64K INTERNALLY INTERLEAVED MEMORY
constant MEM256NIL      equals +%X70  prefix NDT tag $; /*256K NON-INTERLEAVED MEM, LOWER CONTROLLER
constant MEM256EIL      equals +%X71  prefix NDT tag $; /*256K EXTERNALLY INTERLEAVED MFM, LOWER
constant MEM256NIU      equals +%X72  prefix NDT tag $; /*256K NON-INTERLEAVED MEM, UPPER CONTROLLER
constant MEM256EIU      equals +%X73  prefix NDT tag $; /*256K EXTERNALLY INTERLEAVED MEM, UPPER
constant MEM256I        equals +%X74  prefix NDT tag $; /*256K INTERNALLY INTERLEAVED MEMORY

/* BI node device types.  Note low word is hardware device type on BI.
/*  High order word (i.e. the 8000) distinguishes device as a BI device.

/* First BI memory nodes

constant SCORMEM equals +%X80000001  prefix NDT tag $;  /* Scorpio Memory

/* Then other BI devices

constant BIMFA   equals +%X80000101  prefix NDT tag $;  /* BI Multi-Function Adapter
constant BUA     equals +%X80000102  prefix NDT tag $;  /* BI UNIBUS adapter
constant BSA     equals +%X80000104  prefix NDT tag $;  /* BI-SI Adapter
constant KDZ11   equals +%X80000105  prefix NDT tag $;  /* KDZ11 processor
constant NBA     equals +%X80000106  prefix NDT tag $;  /* BI-NMI Adapter
constant BNA     equals +%X80000107  prefix NDT tag $;  /* BI-NI Adapter
constant BCA     equals +%X80000108  prefix NDT tag $;  /* BI-CI Adapter
constant BICOMBO equals +%X80000109  prefix NDT tag $;  /* BI Combo Board
constant BAA     equals +%X8000010A  prefix NDT tag $;  /* BI-VenusAbus Adapter
constant BCI750  equals +%X8000010B  prefix NDT tag $;  /* Interim BI-CI Adapter
constant BIACP   equals +%X8000010C  prefix NDT tag $;  /* Aurora Processor Module
constant AIO     equals +%X8000010D  prefix NDT tag $;  /* Aurora I/O Module
constant BDA     equals +%X8000010E  prefix NDT tag $;  /* BI-to-Disk Adapter
constant AIE     equals +%X8000010F  prefix NDT tag $;  /* Aurora I/O Extension Module

end_module $NDTDEF;
```

```
module $NMBDEF;
/*+
/*
/* FORMAT OF THE FILE NAME BLOCK. THE FILE NAME BLOCK IS USED AS AN INTERNAL
/* INTERFACE TO THE DIRECTORY SCAN ROUTINE, AND IS ALSO THE FORMAT OF A
/* DIRECTORY RECORD.
/*
/*-


aggregate NMBDEF structure prefix NMB$;
    FID_OVERLAY union fill;
        FID word unsigned dimension 3;                  /* FILE ID
        FID_FIELDS structure fill;
            FID_NUM word unsigned;                      /* FID - FILE NUMBER
            FID_SEQ word unsigned;                      /* FID - FILE SEQUENCE NUMBER
            FID_RVN word unsigned;                      /* FID - RELATIVE VOLUME NUMBER
        end FID_FIELDS;
    end FID_OVERLAY;
    NAME word unsigned dimension 3;                     /* FILE NAME (RAD-50)
    TYPE word unsigned;                                 /* FILE TYPE (RAD-50)
    VERSION word;                                       /* VERSION NUMBER
    constant DIRENTRY equals . prefix NMB$ tag K;       /* LENGTH OF DIRECTORY ENTRY
    constant DIRENTRY equals . prefix NMB$ tag C;       /* LENGTH OF DIRECTORY ENTRY

    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                            /* NAME STATUS FLAGS
        FLAGS_BITS structure fill;
            FILL_1 bitfield length 3 fill prefix NMBDEF tag $$;
            ALLVER bitfield mask;                       /* MATCH ALL VERSIONS
            ALLTYP bitfield mask;                       /* MATCH ALL TYPES
            ALLNAM bitfield mask;                       /* MATCH ALL NAMES
            FILL_2 bitfield length 2 fill prefix NMBDEF tag $$;
            WILD bitfield mask;                         /* WILD CARDS IN FILE NAME
            NEWVER bitfield mask;                       /* MAXIMIZE VERSION NUMBER
            SUPERSEDE bitfield mask;                    /* SUPERSEDE EXISTING FILE
            FINDFID bitfield mask;                      /* SEARCH FOR FILE ID
            FILL_3 bitfield length 2 fill prefix NMBDEF tag $$;
            LOWVER bitfield mask;                       /* LOWER VERSION OF FILE EXISTS
            HIGHVER bitfield mask;                      /* HIGHER VERSION OF FILE EXISTS
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    ASCNAME structure fill;
        ASCNAMSIZ byte unsigned;
        ASCNAMTXT character dimension 19;
    end ASCNAME;
    CONTEXT word unsigned;                              /* START POINT FOR NEXT FIND
    constant "LENGTH" equals . prefix NMB$ tag K;       /* LENGTH OF NAME BLOCK
    constant "LENGTH" equals . prefix NMB$ tag C;       /* LENGTH OF NAME BLOCK
end NMBDEF;

end_module $NMBDEF;
```

```
module $NSAARGDEF;

/*+
/* Security Auditing argument list definitions
/*-

/*+
/* Argument list header offset definitions
/*-

aggregate NSAARGHDRDEF structure prefix NSA$;

    ARG_COUNT longword unsigned;                /* Argument list count
    ARG_ID_OVERLAY union fill;
        ARG_ID longword unsigned;               /* Record identification longword
        ARG_ID_FIELDS structure fill;
            ARG_TYPE word unsigned;             /* Record type
            ARG_SUBTYPE word unsigned;          /* Record subtype
        end ARG_ID_FIELDS;
    end ARG_ID_OVERLAY;
    ARG_FLAG_OVERLAY union fill;
        ARG_FLAG byte unsigned;                 /* Flags byte
        FLAG_BITS structure fill;
            ARG_FLAG_ALARM bitfield mask;       /* Generate alarm for this record
            ARG_FLAG_JOURN bitfield mask;       /* Journal this record
            ARG_FLAG_MANDY bitfield mask;       /* Mandatory auditing
        end FLAG_BITS;
    end ARG_FLAG_OVERLAY;
    ARG_PKTNUM byte unsigned;                   /* Number of packets
    ARG_SPARE character length 2;               /* Spare bytes
    ARG_LIST character length 0;

    constant ARGHDR_LENGTH equals . tag C;
    constant ARGHDR_LENGTH equals . tag K;

end NSAARGHDRDEF;


/*+
/* Data packet argument passing mechanism definitions
/*-

constant (ARG_MECH_BYTE,                /* Byte value
          ARG_MECH_WORD,                /* Word value
          ARG_MECH_LONG,                /* Longword value
          ARG_MECH_QUAD,                /* Quadword value
          ARG_MECH_DESCR,               /* Descriptor
          ARG_MECH_ADESCR)              /* Address of descriptor
         equals 0 increment 1 counter #MECHNUM prefix NSA$;

constant ARG_MECHNUM equals #MECHNUM+1 prefix NSA$;


/*+
/* Argument list definitions
/*-
```

```
/* File access

aggregate NSAARG1DEF structure prefix NSA$;

    $$ character length NSA$K_ARGHDR_LENGTH fill; /* Argument list header
    ARG1_FACMOD_TM longword unsigned;           /* FACMOD type and mechanism
    ARG1_FACMOD longword unsigned;              /* File access mode
    ARG1_FILNAM_TM longword unsigned;           /* FILNAM type and mechanism
    ARG1_FILNAM_SIZ longword unsigned;          /* File name size
    ARG1_FILNAM_PTR longword unsigned;          /* File name address
    ARG1_IMGNAM_TM longword unsigned;           /* IMGNAM type and mechanism
    ARG1_IMGNAM quadword unsigned;              /* Image name
    ARG1_PRIVUSED_TM longword unsigned;         /* PRIVUSED type and mechanism
    ARG1_PRIVUSED longword unsigned;            /* Privileges used for access

    constant ARG1_LENGTH equals . tag C;
    constant ARG1_LENGTH equals . tag K;

end NSAARG1DEF;


/* Volume mount

aggregate NSAARG2DEF structure prefix NSA$;

    $$ character length NSA$K_ARGHDR_LENGTH fill; /* Argument list header
    ARG2_UIC_TM longword unsigned;              /* UIC type and mechanism
    ARG2_UIC longword unsigned;                 /* Volume UIC
    ARG2_VOLPRO_TM longword unsigned;           /* VOLPRO type and mechanism
    ARG2_VOLPRO longword unsigned;              /* Volume protection
    ARG2_MOUFLG_TM longword unsigned;           /* MOUFLG type and mechanism
    ARG2_MOUFLG longword unsigned;              /* Mount flags
    ARG2_IMGNAM_TM longword unsigned;           /* IMGNAM type and mechanism
    ARG2_IMGNAM quadword unsigned;              /* Image name
    ARG2_DEVNAM_TM longword unsigned;           /* DEVNAM type and mechanism
    ARG2_DEVNAM_SIZ longword unsigned;          /* Device name size
    ARG2_DEVNAM_PTR longword unsigned;          /* Device name address
    ARG2_LOGNAM_TM longword unsigned;           /* LOGNAM type and mechanism
    ARG2_LOGNAM_SIZ longword unsigned;          /* Logical name size
    ARG2_LOGNAM_PTR longword unsigned;          /* Logical name address
    ARG2_VOLNAM_TM longword unsigned;           /* VOLNAM type and mechanism
    ARG2_VOLNAM_SIZ longword unsigned;          /* Volume name size
    ARG2_VOLNAM_PTR longword unsigned;          /* Volume name address
    ARG2_VOLSNAM_TM longword unsigned;          /* VOLSNAM type and mechanism
    ARG2_VOLSNAM_SIZ longword unsigned;         /* Volume set name size
    ARG2_VOLSNAM_PTR longword unsigned;         /* Volume set name address

    constant ARG2_LENGTH equals . tag C;
    constant ARG2_LENGTH equals . tag K;

end NSAARG2DEF;


/* Volume dismount
```

```
aggregate NSAARG3DEF structure prefix NSA$;

    $$ character length NSA$K_ARGHDR_LENGTH fill; /* Argument list header
    ARG3_DMOUFLG_TM longword unsigned;          /* DMOUFLG type and mechanism
    ARG3_DMOUFLG longword unsigned;             /* Dismount flags
    ARG3_IMGNAM_TM longword unsigned;           /* IMGNAM type and mechanism
    ARG3_IMGNAM quadword unsigned;              /* Image name
    ARG3_DEVNAM_TM longword unsigned;           /* DEVNAM type and mechanism
    ARG3_DEVNAM_SIZ longword unsigned;          /* Device name size
    ARG3_DEVNAM_PTR longword unsigned;          /* Device name address
    ARG3_LOGNAM_TM longword unsigned;           /* LOGNAM type and mechanism
    ARG3_LOGNAM_SIZ longword unsigned;          /* Logical name size
    ARG3_LOGNAM_PTR longword unsigned;          /* Logical name address
    ARG3_VOLNAM_TM longword unsigned;           /* VOLNAM type and mechanism
    ARG3_VOLNAM_SIZ longword unsigned;          /* Volume name size
    ARG3_VOLNAM_PTR longword unsigned;          /* Volume name address
    ARG3_VOLSNAM_TM longword unsigned;          /* VOLSNAM type and mechanism
    ARG3_VOLSNAM_SIZ longword unsigned;         /* Volume set name size
    ARG3_VOLSNAM_PTR longword unsigned;         /* Volume set name address

    constant ARG3_LENGTH equals . tag C;
    constant ARG3_LENGTH equals . tag K;

end NSAARG3DEF;

end_module $NSAARGDEF;
```

```
module $NSAEVTDEF;

/*+
/* Security Auditing event class bit definitions: This macro defines
/* the bits which are used to enable audit journaling and alarms for
/* each class of system event.
/*-

aggregate NSAEVTDEF structure prefix NSA$;

    EVT_SYS_OVERLAY union fill;
        EVT_SYS longword unsigned;      /* Misc system event mask
        EVT_SYS_BITS structure fill;
            EVT_ACL bitfield mask;      /* ACL requested audits
            EVT_MOUNT bitfield mask;    /* MOUNT and DISMOUNT requests
                                        /* Modifications made to the system
            EVT_UAF bitfield mask;      /* or network authorization files
            EVT_SPARE bitfield length 32-^ mask;
        end EVT_SYS_BITS;
    end EVT_SYS_OVERLAY;

    EVT_LOGB byte unsigned;             /* Breakin detection event mask
    EVT_LOGI byte unsigned;             /* Login  event mask
    EVT_LOGF byte unsigned;             /* Login failure event mask
    EVT_LOGO byte unsigned;             /* Logout event mask
/*****************************************************************************
/* The following file access masks must be contiguous and in the current order
/*****************************************************************************
    EVT_FAILURE longword unsigned;      /* Access failures event mask
    EVT_SUCCESS longword unsigned;      /* Successful access event mask
    EVT_SYSPRV longword unsigned;       /* Success due to SYSPRV event mask
    EVT_BYPASS longword unsigned;       /* Success due to BYPASS event mask
    EVT_UPGRADE longword unsigned;      /* Success due to UPGRADE event mask
    EVT_DOWNGRADE longword unsigned;    /* Success due to DOWNGRADE event mask
    EVT_GRPPRV longword unsigned;       /* Success due to GRPPRV event mask
    EVT_READALL longword unsigned;      /* Success due to READALL event mask
/*****************************************************************************
/* End of file access masks
/*****************************************************************************

    constant EVT_LENGTH equals . tag C;
    constant EVT_LENGTH equals . tag K;

end NSAEVTDEF;

aggregate NSAEVTLOGBITS structure prefix NSA$;
    EVT_LOG_BAT bitfield mask;          /* Batch
    EVT_LOG_DIA bitfield mask;          /* Dialup
    EVT_LOG_LOC bitfield mask;          /* Local
    EVT_LOG_REM bitfield mask;          /* Remote
    EVT_LOG_NET bitfield mask;          /* Network
    EVT_LOG_SUB bitfield mask;          /* Subprocess
    EVT_LOG_DET bitfield mask;          /* Detached process
end NSAEVTLOGBITS;

end_module $NSAEVTDEF;
```

```
end
end_
```

```
module $NSAIDTDEF;

/*+
/* Security Auditing Impure Data Table offset definitions
/*-

aggregate NSAIDTDEF structure prefix NSA$;

     IDT_ALARM_HDR character length 38+8;        /* Alarm header buffer
     IDT_RECORD_BUF character length 1024;       /* Record buffer
     IDT_RECORD_DESCR quadword unsigned;         /* Record buffer descriptor
     IDT_RECORD_DT character length 128;         /* Record descriptor table
     IDT_AUDIT_CHAN longword unsigned;           /* Audit journal channel number

     constant IDT_LENGTH equals . tag C;
     constant IDT_LENGTH equals . tag K;

     constant IDT_PAGES equals (.+511)@-9;       /* Number of pages for IDT

end NSAIDTDEF;

end_module $NSAIDTDEF;
```

```
module $ORBDEF;
/*+
/*
/* Object's Rights Block - structure defining the protection information
/* for various objects within the system.
/*
/*-

aggregate ORBDEF structure prefix ORB$;
    OWNER structure longword unsigned;           /* Object's owner
        UICMEMBER word unsigned;                 /* Member number
        UICGROUP word unsigned;                  /* Group number
    end OWNER;
    ACL_MUTEX longword unsigned;                 /* Mutex for this ACL
    SIZE word unsigned;                          /* Size of the ORB in bytes
    TYPE byte unsigned;                          /* Structure type
    FLAGS structure byte unsigned;               /* Field modifiers
        PROT_16 bitfield mask;                   /* Use word not vector protection
        ACL_QUEUE bitfield mask;                 /* Use ACL queue not descriptor list
        MODE_VECTOR bitfield mask;               /* Use vector not byte mode protection
        NOACL bitfield mask;                     /* Object cannot have an ACL
        CLASS_PROT bitfield mask;                /* Security classification is valid
    end FLAGS;
    FILL_1 word fill;                            /* Unused
    REFCOUNT word unsigned;                      /* Reference count
    MODE_OVERLAY union fill;
        MODE_PROT structure quadword unsigned;   /* Mode protection vector
            MODE_PROTL longword unsigned;        /* Low longword of vector
            MODE_PROTH longword unsigned;        /* High longword of vector
        end MODE_PROT;
        MODE byte unsigned;                      /* Simple access mode
    end MODE_OVERLAY;
    SYS_PROT_OVERLAY union fill;
        SYS_PROT longword unsigned;              /* System protection field
        PROT word unsigned;                      /* Standard SOGW protection
    end SYS_PROT_OVERLAY;
    OWN_PROT longword unsigned;                  /* Owner protection field
    GRP_PROT longword unsigned;                  /* Group protection field
    WOR_PROT longword unsigned;                  /* World protection field
    ACL_1_OVERLAY union fill;
        ACLFL longword unsigned;                 /* ACL queue forward link
        ACL_COUNT longword unsigned;             /* Count of ACL segments
    end ACL_1_OVERLAY;
    ACL_2_OVERLAY union fill;
        ACLBL longword unsigned;                 /* ACL queue backward link
        ACL_DESC longword unsigned;              /* Address of ACL segment descriptor list
    end ACL_2_OVERLAY;
    MIN_CLASS structure;
        FILL_2 byte dimension 20 fill;           /* Minimum classification mask
    end MIN_CLASS;
    MAX_CLASS structure;
        FILL_3 byte dimension 20 fill;           /* Maximum classification mask
    end MAX_CLASS;
    constant "LENGTH" equals . prefix ORB$ tag K;        /* Structure length
    constant "LENGTH" equals . prefix ORB$ tag C;        /* Structure length
end ORBDEF;
```

```
end_module $ORBDEF;
```

```
module $PBDEF;
/*+
/* PB - SCS PATH BLOCK
/*
/* THE PB HAS INFORMATION ABOUT THE PHYSICAL PATH TO ANOTHER
/* SYSTEM IN A CLUSTER.  PATH BLOCKS TO THE SAME SYSTEM ARE
/* LINKED TOGETHER TO THE SYSTEM BLOCK (SB).
/*-


aggregate PBDEF structure prefix PB$;
    FLINK longword unsigned;                                /*FWD LINK TO NEXT PB
    BLINK longword unsigned;                                /*BACK LINK TO PREVIOUS PB
    SIZE word unsigned;                                     /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                                     /*SCS STRUCTURE TYPE
    SUBTYP byte unsigned;                                   /*SCS STRUCT SUBTYPE FOR PB
    RSTATION byte unsigned dimension 6;                     /*REMOTE STATION ADDRESS
    STATE word unsigned;                                    /*PATH STATE
                                                            /*STATE DEFINITIONS:
                                                            /* 0 ORIGIN, INCREMENTS OF 1
    constant(
          CLOSED                                            /* NEWLY CREATED PATHBLOCK
        , ST_SENT                                           /* START SENT
        , ST_REC                                            /* START RECEIVED
        , OPEN                                              /* OPEN PORT-PORT VIRTUAL CIRCUIT
        ) equals 0 increment 1  prefix PB tag $C;           /*
    constant VC_FAIL    equals (%X8000)  prefix PB tag $C;/* VC FAILURE IN PROGRESS STATE
    constant PWR_FAIL   equals (%X4000)  prefix PB tag $C;/* PWR FAIL RECOVERY IN PROGRESS STATE
    RPORT_TYP_OVERLAY union fill;
        RPORT_TYP longword unsigned;                        /*HARDWARE PORT TYPE CODE
        RPORT_TYP_BITS structure fill;
            PORT_TYP bitfield length 31;                    /* HARDWARE PORT TYPE,
            DUALPATH bitfield mask;                         /* 0/1 FOR SINGLE PATH/DUAL PATH PORT
        end RPORT_TYP_BITS;
    end RPORT_TYP_OVERLAY;
    constant CI780      equals 2        prefix PB tag $C;/* CI780 PORT
    constant CI750      equals 2        prefix PB tag $C;/* CI750 PORT (=CI780)
    constant HSC        equals 4        prefix PB tag $C;/* HSC PORT
    constant KL10       equals 6        prefix PB tag $C;/* KLIPA PORT
    constant CINT       equals 7        prefix PB tag $C;/* CI NODE TESTER
    constant NI         equals 8        prefix PB tag $C;/* NI-DEUNA PORT
    constant PS         equals 9        prefix PB tag $C;/* PASSTHRU PORT

    RPORT_REV longword unsigned;                            /*REMOTE PORT HW REV LEVEL
    RPORT_FCN longword unsigned;                            /*REMOTE PORT FUNCTION MASK
    RST_PORT byte unsigned;                                 /*OWNING PORT WHICH RESET REMOTE PORT
    RSTATE_OVERLAY union fill;
        RSTATE byte unsigned;                               /*REMOTE PORT STATUS:
        RSTATE_BITS structure fill;
            MAINT bitfield mask;                            /* 0/1 FOR MAINTENANCE MODE NO/YES
            STATE bitfield length 2;                        /* REMOTE PORT STATE:
        end RSTATE_BITS;
                                                            /* DEFINE REMOTE STATES, 0 ORIGIN
        constant(
              UNINIT                                        /* UNINITIALIZED,
            , DISAB                                          /* DISABLED
            , ENAB                                           /* ENABLED
            } equals 0 increment 1  prefix PB tag $C;        /*
    end RSTATE_OVERLAY;
    RETRY word unsigned;                                     /*START HANDSHAKE RETRY COUNT
    LPORT_NAME character length 4;                           /*LOCAL PORT DEVICE NAME
    CBL_STS_OVERLAY union fill;
        CBL_STS byte unsigned;                               /*CABLE STATUS TO THE REMOTE
        CBL_STS_BITS structure fill;
            CUR_CBL bitfield mask;                           /* 1/0 FOR CURRENT STATUS OK/BAD
        end CBL_STS_BITS;
    end CBL_STS_OVERLAY;
    P0_STS byte unsigned;                                    /*PATH 0 STATUS
    P1_STS_OVERLAY union fill;
        P1_STS byte unsigned;                                /*PATH 1 STATUS
        P1_STS_BITS structure fill;
            CUR_PS bitfield mask;                            /* 1/0 FOR CURRENT STATUS OK/BROKEN
        end P1_STS_BITS;
    end P1_STS_OVERLAY;
    FILL_1 byte fill prefix PBDEF tag $$;                    /*RESERVED BYTE
    PDT longword unsigned;                                   /*ADDR OF PORT DESCRIPTOR TABLE FOR
                                                             /* LOCAL PORT
    SBLINK longword unsigned;                                /*LINK TO SYSTEM BLOCK
    CDTLST longword unsigned;                                /*LINK TO FIRST CDT OVER THIS PATH
                                                             /* (0 IF NO CDT'S)
    WAITQFL longword unsigned;                               /* SCS SEND MSG WAIT QUEUE FLINK
    WAITQBL_OVERLAY union fill;
        WAITQBL longword unsigned;                           /*SCS SEND MSG WAIT QUEUE BLINK
        DUETIME longword unsigned;                           /*START HANDSHAKE TIMER
    end WAITQBL_OVERLAY;
    SCSMSG longword unsigned;                                /*ADDR OF SCS MESSAGE BUFFER
    STS_OVERLAY union fill;
        STS word unsigned;                                   /*PATH BLOCK STATUS
        STS_BITS structure fill;
            TIM bitfield mask;                               /* HANDSHAKE TIMEOUT IN PROGRESS
        end STS_BITS;
    end STS_OVERLAY;
    VCFAIL_RSN word unsigned;                                /*VC FAILURE REASON (VMS
                                                             /*STATUS CODE
    PROTOCOL byte unsigned;                                  /*PPD PROTOCOL LEVEL
    FILL_2 byte dimension 3 fill prefix PBDEF tag $$;        /*RESERVED BYTES
    FILL_3 longword dimension 2 fill prefix PBDEF tag $$;/*RESERVED LONGWDS
    constant "LENGTH" equals . prefix PB$ tag K;             /*LENGTH OF A PATH BLOCK
    constant "LENGTH" equals . prefix PB$ tag C;             /*LENGTH OF A PATH BLOCK

end PBDEF;

end_module $PBDEF;
```

```
module $PBHDEF;
/*+
/* DEFINE PERFORMANCE BUFFER HEADER
/*-


aggregate PBHDEF structure prefix PBH$;
    BUFRFL longword unsigned;                             /*BUFFER FORWARD LINK
    BUFRBL longword unsigned;                             /*BUFFER BACKWARD LINK
    SIZE word unsigned;                                   /*SIZE OF PERFORMANCE DATA BUFFER
    TYPE byte unsigned;                                   /*DATA STRUCTURE TYPE
    MSGCNT word unsigned;                                 /*COUNT OF MESSAGES IN BUFFER
    constant START equals . prefix PBH$ tag K;            /*START OF DATA AREA
    constant START equals . prefix PBH$ tag C;            /*START OF DATA AREA
    FILL_1 byte dimension 499 fill prefix PBHDEF tag $$;/*DATA AREA
    constant "LENGTH" equals . prefix PBH$ tag K;         /*LENGTH OF PERFORMANCE DATA BUFFER
    constant "LENGTH" equals . prefix PBH$ tag C;         /*LENGTH OF PERFORMANCE DATA BUFFER

end PBHDEF;

end_module $PBHDEF;
```

```
module $PBODEF;
/*+
/* PBO - SCS$CONFIG_PTH CALL OUTPUT ARRAY FORMAT
/*
/* THE OUTPUT ARRAY RETURNED FROM THE SCS$CONFIG_PTH CALL.  DATA IS MOSTLY COPIED
/* FROM THE PATH BLOCK (PB) BEING LOOKED UP.
/*-


aggregate PBODEF structure prefix PBO$;
    RSTATION byte unsigned dimension 6;                 /*REMOTE STATION ADDR
    STATE word unsigned;                                /*PATH STATE
    RPORT_TYP longword unsigned;                        /*REMOTE PORT HW PORT TYPE
    RPORT_REV longword unsigned;                        /*REMOTE PORT REV LEVEL
    RPORT_FCN longword unsigned;                        /*REMOTE PORT FUNCTION MASK
    RST_PORT byte unsigned;                             /*OWNING PORT WHICH LAST
                                                        /* RESET THIS REMOTE
    RSTATE byte unsigned;                               /*REMOTE PORT STATE
    RETRY word unsigned;                                /*START HANDSHAKE RETRIES LEFT
    LPORT_NAME character length 4;                      /*LOCAL PORT DEVICE NAME
    CBL_STS byte unsigned;                              /*CURRENT CABLE STATUS
    P0_STS byte unsigned;                               /*PATH 0 STATUS
    P1_STS byte unsigned;                               /*PATH 1 STATUS
    FILL_1 byte fill prefix PBODEF tag $$;              /*RESERVED BYTE
    constant NXT_VC equals . prefix PBO$ tag C;         /*SPECIFIER OF NEXT VC (PB)
    constant NXT_VC equals . prefix PBO$ tag K;         /* TO THIS SYSTEM (12 BYTE
                                                        /* SPECIFIER FOLLOWS:)
    NXT_RSTAT byte unsigned dimension 6;                /* REMOTE STATION ADDR
    FILL_1 word fill prefix PBODEF tag $$;              /* RESERVED WORD
    NXT_LPORT character length 4;                       /* LOCAL PORT NAME ON NXT PB
    SYSTEMID byte unsigned dimension 6;                 /*ID OF SYSTEM ASSOC WITH
                                                        /* THIS PB
    FILL_1 word fill prefix PBODEF tag $$;              /*RESERVED WORD
    constant "LENGTH" equals . prefix PBO$ tag C;       /*LENGTH OF PBO
    constant "LENGTH" equals . prefix PBO$ tag K;       /*LENGTH OF PBO

end PBODEF;

end_module $PBODEF;
```

```
module $PCBDEF;
/*+
/* PCB DEFINITIONS
/*-


aggregate PCBDEF structure prefix PCB$;
    SQFL longword unsigned;                             /*STATE QUEUE FORWARD LINK
    SQBL longword unsigned;                             /*STATE QUEUE BACKWARD LINK
    SIZE word unsigned;                                 /*SIZE IN BYTES
    TYPE byte unsigned;                                 /*STRUCTURE TYPE CODE FOR PCB
    PRI byte unsigned;                                  /*PROCESS CURRENT PRIORITY
    ASTACT byte unsigned;                               /*ACCESS MODES WITH ACTIVE ASTS
    ASTEN byte unsigned;                                /*ACCESS MODES WITH ASTS ENABLED
    MTXCNT word unsigned;                               /*COUNT OF MUTEX SEMAPHORES OWNED
    ASTQFL longword unsigned;                           /*AST QUEUE FORWARD LINK(HEAD)
    ASTQBL longword unsigned;                           /*AST QUEUE BACK LINK(TAIL)
    PHYPCB longword unsigned;                           /*PHYSICAL ADDRESS OF HW PCB
    OWNER longword unsigned;                            /*PID OF CREATOR
    WSSWP longword unsigned;                            /*SWAP FILE DISK ADDRESS
    STS structure longword unsigned;                    /*PROCESS STATUS FLAGS
        RES bitfield mask;                              /* RESIDENT, IN BALANCE SET
        DELPEN bitfield mask;                           /* DELETE PENDING
        FORCPEN bitfield mask;                          /* FORCE EXIT PENDING
        INQUAN bitfield mask;                           /* INITIAL QUANTUM IN PROGRESS
        PSWAPM bitfield mask;                           /* PROCESS SWAP MODE (1=NOSWAP)
        RESPEN bitfield mask;                           /* RESUME PENDING, SKIP SUSPEND
        SSFEXC bitfield mask;                           /* SYSTEM SERVICE EXCEPTION ENABLE (K)
        SSFEXCE bitfield mask;                          /* SYSTEM SERVICE EXCEPTION ENABLE (E)
        SSFEXCS bitfield mask;                          /* SYSTEM SERVICE EXCEPTION ENABLE (S)
        SSFEXCU bitfield mask;                          /* SYSTEM SERVICE EXCEPTION ENABLE (U)
        SSRWAIT bitfield mask;                          /* SYSTEM SERVICE RESOURCE WAIT DISABLE
        SUSPEN bitfield mask;                           /* SUSPEND PENDING
        WAKEPEN bitfield mask;                          /* WAKE PENDING, SKIP HIBERNATE
        WALL bitfield mask;                             /* WAIT FOR ALL EVENTS IN MASK
        BATCH bitfield mask;                            /* PROCESS IS A BATCH JOB
        NOACNT bitfield mask;                           /* NO ACCOUNTING FOR PROCESS
        SWPVBN bitfield mask;                           /* WRITE FOR SWP VBN IN PROGRESS
        ASTPEN bitfield mask;                           /* AST PENDING
        PHDRES bitfield mask;                           /* PROCESS HEADER RESIDENT
        HIBER bitfield mask;                            /* HIBERNATE AFTER INITIAL IMAGE ACTIVATE
        LOGIN bitfield mask;                            /* LOGIN WITHOUT READING AUTH FILE
        NETWRK bitfield mask;                           /* NETWORK CONNECTED JOB
        PWRAST bitfield mask;                           /* POWER FAIL AST
        NODELET bitfield mask;                          /* NO DELETE
        DISAWS bitfield mask;                           /* 1=DISABLE AUTOMATIC WS ADJUSTMENT
        INTER bitfield mask;                            /* PROCESS IS AN INTERACTIVE JOB
        RECOVER bitfield mask;                          /* PROCESS CAN RECOVER LOCKS
        SECAUDIT bitfield mask;                         /* MANDATORY SECURITY AUDITING
    end STS;
    WTIME structure longword unsigned;                  /*TIME AT START OF WAIT
        PRISAV byte unsigned;                           /*SAVED CURRENT PRIORITY
        PRIBSAV byte unsigned;                          /*SAVE BASE PRIORITY
        DPC byte unsigned;                              /*DELETE PENDING COUNT
        AUTHPRI byte unsigned;                          /*INITIAL PROCESS PRIORITY
    end WTIME;
    STATE word unsigned;                                        /*PROCESS STATE
    WEFC byte unsigned;                                         /*WAITING EF CLUSTER NUMBER
    PRIB byte unsigned;                                         /*BASE PRIORITY
    APTCNT word unsigned;                                       /*ACTIVE PAGE TABLE COUNT
    TMBU word unsigned;                                         /*TERMINATION MAILBOX UNIT NO.
    GPGCNT word unsigned;                                       /*GLOBAL PAGE COUNT IN WS
    PPGCNT word unsigned;                                       /*PROCESS PAGE COUNT IN WS
    ASTCNT word unsigned;                                       /*AST COUNT REMAINING
    BIOCNT word unsigned;                                       /*BUFFERED I/O COUNT REMAINING
    BIOLM word unsigned;                                        /*BUFFERED I/O LIMIT
    DIOCNT word unsigned;                                       /*DIRECT I/O COUNT REMAINING
    DIOLM word unsigned;                                        /*DIRECT I/O COUNT LIMIT
    PRCCNT word unsigned;                                       /*SUBPROCESS COUNT
    TERMINAL character length 8;                                /*TERMINAL DEVICE NAME STRING
                                                                /*FOR INTERACTIVE JOBS

    PQB_OVERLAY union fill;
        PQB longword unsigned;                                  /*POINTER TO PROCESS QUOTA BLOCK
                                                                /*(PROCESS CREATION ONLY)
        EFWM longword unsigned;                                 /*EVENT FLAG WAIT MASK
    end PQB_OVERLAY;
    EFCS longword unsigned;                                     /*LOCAL EVENT FLAG CLUSTER,SYSTEM
    EFCU longword unsigned;                                     /*LOCAL EVENT FLAG CLUSTER, USER
    CEFC_OVERLAY union fill;
        CEFC_OVERLAY_1 structure fill;                  /*(USED BY SHELL)
            PGFLCHAR word unsigned;                     /*PAGE FILE CHARACTERISTICS
            PGFLINDEX byte unsigned;                    /*DESIRED PAGE FILE INDEX
            PGFL_FILL_1 byte fill tag $$;               /*SPARE
            SWAPSIZE longword unsigned;                 /*INITIAL SWAP BLOCK ALLOCATION
            end CEFC_OVERLAY_1;
        CEFC_OVERLAY_2 structure fill;
            EFC2P longword unsigned;                    /*POINTER TO GLOBAL CLUSTER !2
            EFC3P longword unsigned;                    /*POINTER TO GLOBAL CLUSTER !3
            end CEFC_OVERLAY_2;
        end CEFC_OVERLAY;
    PID longword unsigned;                                      /*PROCESS ID USED BY EXEC ON LOCAL NODE ONLY
/*
/**** WARNING - THE INTERNAL STRUCTURE OF THE EPID IS SUBJECT TO RADICAL CHANGE BETWEEN
/****           VERSIONS OF VMS.  NO ASSUMPTIONS SHOULD EVER BE MADE ABOUT ITS FORMAT
/*
    EPID structure longword unsigned;                           /*CLUSTER-WIDE PROCESS ID SEEN BY THE WORLD
        EPID_PROC bitfield length 21;                           /*PROCESS ID FIELD, CAN CONVERT TO PCB$L_PID
{
{       Currently, the PCB$V_EPID_PROC field can be decomposed into the PCB$L_PID by extracting the
{       process index and sequence number according to:
{
{           EPID_PROC_PIX  bitfield length SCH$GL_PIXWIDTH;
{           EPID_PROC_SEQ  bitfield length (PCB$S_EPID_PROC - SCH$GL_PIXWIDTH);
{
        EPID_NODE_IDX  bitfield length 8;                       /*IDX - INDEX TO TABLE OF NODE IDENTIFICATIONS
        EPID_NODE_SEQ  bitfield length 2;                       /*SEQ - SEQUENCE NUMBER FOR NODE TABLE ENTRY REUSE
        EPID_WILD  bitfield mask;                               /*FLAG THAT EPID IS WILDCARD CONTEXT FOR $GETJPI, AND NOT
        end EPID;                                               /* A VALID EPID
/*
    EOWNER longword unsigned;                                   /*EPID OF PROCESS OWNER
/*
    PHD longword unsigned;                                      /*PROCESS HEADER ADDRESS
    LNAME character length 16;                          /*LOGICAL NAME OF PROCESS
    JIB longword unsigned;                              /*ADDRESS OF JOB INFORMATION BLOCK
    PRIV quadword unsigned;                             /*CURRENT PRIVILEGE MASK
    ARB longword unsigned;                              /*ADDRESS OF ACCESS RIGHTS BLOCK
    ARB_FILL_1 byte dimension 44 fill tag $$;           /*RIGHTS LIST DESCRIPTORS, ETC.
    UIC structure longword unsigned;                    /*LOGON UIC OF PROCESS
        MEM word unsigned;                              /*MEMBER NUMBER IN UIC
        GRP word unsigned;                              /*GROUP NUMBER IN UIC
    end UIC;
    ARB_FILL_2 byte dimension 60 fill tag $$;           /*REMAINDER OF ARB
    ACLFL longword unsigned;                            /* ACL queue forward link
    ACLBL longword unsigned;                            /* ACL queue backward link
    LOCKQFL longword unsigned;                          /*LOCK QUEUE FORWARD LINK
    LOCKQBL longword unsigned;                          /*LOCK QUEUE BACKWARD LINK
    DLCKPRI longword unsigned;                          /*DEADLOCK RESOLUTION PRIORITY
    IPAST longword unsigned;                            /*VECTOR OF MODE BITS FOR IPASTS
    DEFPROT longword unsigned;                          /*PROCESS DEFAULT PROTECTION
    WAITIME longword unsigned;                          /*ABS TIME OF LAST PROCESS EVENT
    PMB longword unsigned;                              /*PMB ADDRESS
    constant "LENGTH" equals . prefix PCB$ tag K;       /*LENGTH OF PCB
    constant "LENGTH" equals . prefix PCB$ tag C;       /*LENGTH OF PCB

end PCBDEF;

end_module $PCBDEF;
```

```
module $PDBDEF;
/*+
/* DEFINE DEVICE PERFORMANCE DATA BLOCK
/*-


aggregate PDBDEF structure prefix PDB$;
    FREEFL longword unsigned;                         /*FREE BUFFER LISTHEAD FORWARD LINK
    FREEBL longword unsigned;                         /*FREE BUFFER LISTHEAD BACKLINK
    SIZE word unsigned;                               /*SIZE OF DATA STRUCTURE
    TYPE byte unsigned;                               /*TYPE OF DATA STRUCTURE
    OVERRUN byte unsigned;                            /*OVERRUN INDICATOR
    FILLFL longword unsigned;                         /*FILLED BUFFER LISTHEAD FORWARD LINK
    FILLBL longword unsigned;                         /*FILLED BUFFER LISTHEAD BACKWARD LINK
    CURBUF longword unsigned;                         /*ADDRESS OF CURRENT BUFFER
    NXTBUF longword unsigned;                         /*ADDRESS OF NEXT LOCATION IN BUFFER
    ENDBUF longword unsigned;                         /*ADDRESS OF END OF BUFFER
    PID longword unsigned;                            /*PROCESS ID OF DATA COLLECTION PROCESS
    DEVCLASS byte unsigned;                           /*DEVICE CLASS SELECTION
    DEVTYPE byte unsigned;                            /*DEVICE TYPE SELECTION
    ANDM word unsigned;                               /*STATUS SELECTION 'AND' MASK
    XORM word unsigned;                               /*STATUS SELECTION 'XOR' MASK
    BUFCNT word unsigned;                             /*COUNT OF FILLED BUFFERS
    FUNC quadword unsigned;                           /*SELECTION FUNCTION MASK
    constant "LENGTH" equals . prefix PDB$ tag K;     /*LENGTH OF DATA CONTROL BLOCK
    constant "LENGTH" equals . prefix PDB$ tag C;     /*LENGTH OF DATA CONTROL BLOCK

end PDBDEF;

end_module $PDBDEF;
```

```
module $PDTDEF;
/*+
/* DEFINE PORT-INDEPENDENT OFFSETS IN A PORT DESCRIPTOR TABLE.
/*
/* THERE IS ONE PDT PER PORT ACCESSED VIA SCS.  THESE PORTS INCLUDE
/* CI'S AND UDA'S.  THE PDT CONTAINS A PORT-INDEPENDENT PIECE (DEFINED
/* HERE) FOLLOWED BY AN OPTIONAL PORT-SPECIFIC PIECE DEFINED IN THE
/* PORT DRIVER.  PDT'S ARE CREATED BY THE CONTROLLER INIT ROUTINES
/* OF THE INDIVIDUAL PORT DRIVERS.
/*-


aggregate PDTDEF structure prefix PDT$;
    FLINK longword unsigned;                          /*LINK TO NEXT SCS PDT
    PORTCHAR_OVERLAY union fill;
        PORTCHAR word unsigned;                       /*Port Characteristics
        PORTCHAR_BITS structure fill;
            SNGLHOST bitfield mask;                   /* Port to single host bus
        end PORTCHAR_BITS;
    end PORTCHAR_OVERLAY;
    FILL_2 byte fill prefix PDTDEF tag $$;            /* UNUSED BYTE
    PDT_TYPE byte unsigned;                           /* TYPE OF PDT
    constant PA equals 1  prefix PDT tag $C;          /* CI PORT
    constant PU equals 2  prefix PDT tag $C;          /* UDA PORT
    constant PE equals 3  prefix PDT tag $C;          /* NI PORT
    constant PS equals 4  prefix PDT tag $C;          /* PASSTHRU PORT
    SIZE word unsigned;                               /*STRUCTURE SIZE IN BYTES
    TYPE byte unsigned;                               /*STRUCTURE TYPE = SCS
    SUBTYP byte unsigned;                             /*STRUCTURE SUBTYPE
    constant SCSBASE equals . prefix PDT$ tag K;      /*SCS ENTRIES INTO THE PORT DRIVER:
    constant SCSBASE equals . prefix PDT$ tag C;      /*SCS ENTRIES INTO THE PORT DRIVER:
    ACCEPT longword unsigned;                         /* ACCEPT A CONNECT REQUEST
    ALLOCDG longword unsigned;                        /* ALLOCATE A DG BUFFER
    ALLOCMSG longword unsigned;                       /* ALLOCATE A MESSAGE BUFFER
    CONNECT longword unsigned;                        /* REQUEST CONNECTION TO REMOTE
    DEALLOCDG longword unsigned;                      /* DEALLOCATE DG BUFFER
    DEALLOMSG longword unsigned;                      /* DEALLOCATE MSG BUFFER
    DEALRGMSG longword unsigned;                      /* DEALLOC MSG BUFF, ARGS IN REGISTERS
    DCONNECT longword unsigned;                       /* BREAK CONNECTION
    MAP longword unsigned;                            /* MAP A BUFFER FOR BLK XFER
    MAPBYPASS longword unsigned;                      /* MAP, DISABL ACCESS CHECKS
    MAPIRP longword unsigned;                         /* MAP, GET ARGS FROM IRP
    MAPIRPBYP longword unsigned;                      /* MAP, ARGS FROM IRP, DISABL ACCESS CHECKS
    QUEUEDG longword unsigned;                        /* QUEUE A DG FOR RECEIVE
    QUEUEMDGS longword unsigned;                      /* ALLOC/DEALLOC DG'S FOR RECEIVE
    RCHMSGBUF longword unsigned;                      /* RECYCLE MSG BUFF, HIGH PRIORITY
    RCLMSGBUF longword unsigned;                      /* RECYCLE MSG BUFF, LOW PRIORITY
    REJECT longword unsigned;                         /* REJECT CONNECT REQUEST
    REQDATA longword unsigned;                        /* REQUEST BLK DATA XFER
    SENDDATA longword unsigned;                       /* SEND BLK DATA XFER
    SENDDG longword unsigned;                         /* SEND A DATAGRAM
    SENDMSG longword unsigned;                        /* SEND A MESSAGE
    SNDCNTMSG longword unsigned;                      /* SEND MSG WITH BYTE COUNT
    UNMAP longword unsigned;                          /* UNMAP A BUFFER
    READCOUNT longword unsigned;                      /* READ COUNTERS (FMT PORT SPECIFIC)
    RLSCOUNT longword unsigned;                         /* RELEASE AND READ COUNTERS
    MRESET longword unsigned;                           /* MAINT RESET OF REMOTE
    MSTART longword unsigned;                           /* MAINT START OF REMOTE
    MAINTFCN longword unsigned;                         /* MISC MAINT FUNCTIONS NOT SUPPORTED
                                                        /*   IN VMS
    SENDRGDG longword unsigned;                         /* SEND DG W/ REGISTER INPUTS
    STOP_VCS longword unsigned;                         /* SEND STOP DGS ON ALL VCS
    constant SCSEND equals . prefix PDT$ tag K;         /*END OF SCS ENTRIES TO PORT DRIVER
    constant SCSEND equals . prefix PDT$ tag C;         /*END OF SCS ENTRIES TO PORT DRIVER
    FILL_3 longword dimension 10 fill prefix PDTDEF tag $$; /*RESERVED VECTORS
    WAITQFL longword unsigned;                          /*LISTHEAD FOR FORK BLOCKS WAITING
    WAITQBL longword unsigned;                          /* FOR NONPAGED POOL
    MSGHDRSZ longword unsigned;                         /*MESSAGE HEADER SIZE
    DGOVRHD longword unsigned;                          /*DATAGRAM HEADER SIZE
    MAXBCNT longword unsigned;                          /*MAXIMUM TRANSFER BCNT
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                            /*PORT FLAGS
        FLAGS_BITS structure fill;
            CNTBSY bitfield mask;                       /* COUNTERS IN USE
            CNTRLS bitfield mask;                       /* RELEASE COUNTERS
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    FILL_4 word fill prefix PDTDEF tag $$;              /*RESERVED WORD
    CNTOWNER character length 16;                       /*NAME OF SYSAP USING COUNTERS
    CNTCDRP longword unsigned;                          /*CDRP OF SYSAP READING COUNTERS
    POLLSWEEP longword unsigned;                        /*# SECONDS TO DO A POLLER SWEEP
    UCB0 longword unsigned;                             /*ADDR OF UCB.
    ADP  longword unsigned;                             /*ADDR OF ADP.
    constant "LENGTH" equals . prefix PDT$ tag K;       /*SIZE OF PORT-INDEPENDENT PIECE
    constant "LENGTH" equals . prefix PDT$ tag C;       /*SIZE OF PORT-INDEPENDENT PIECE
                                                        /* OF PDT.

end PDTDEF;

end_module $PDTDEF;
```

```
module $PFBDEF;

/*+
/* PAGE FAULT MONITOR BUFFER
/*-


aggregate PFBDEF structure prefix PFB$;
    FLINK longword unsigned;                            /*Forward link
    BLINK longword unsigned;                            /*Back    link
    SIZE word unsigned;                                 /*Structure size
    TYPE byte unsigned;                                 /*Dynamic structure type (PFB)
    SPARE_1 byte fill prefix PFBDEF tag $$;             /*SPARE

    #pfb_ubuff_size = 512;
    constant 'USER_BUFFER' equals . prefix PFB$ tag B;
    USER_BUFFER structure;                              /*Buffer returned to user
        #pfb_ubase = .;
        RECCNT longword unsigned;                       /*Record count
        OVERFLOW longword unsigned;                     /*Overflow count
        #pfb_ubuff_oh = . - #pfb_ubase;
        constant 'BUFFER' equals . prefix PFB$ tag B;   /*Beginning of PC/VA pairs
        FILL_1 byte dimension (#pfb_ubuff_size - #pfb_ubuff_oh) fill prefix PFBDEF tag $$;
    end USER_BUFFER;

    constant 'LENGTH' equals . prefix PFB$ tag K;       /*Length of PFB
    constant 'LENGTH' equals . prefix PFB$ tag C;       /*Length of PFB
end PFBDEF;

end_module $PFBDEF;
```

```
module $PFLDEF;

/*+
/* PAGE FILE CONTROL BLOCK
/*-


/*
/* ***** L_VBN, L_WINDOW, AND B_PFC MUST BE THE SAME OFFSET VALUES AS THE
/* ***** EQUIVALENTLY NAMED OFFSETS IN $SECDEF
/*

aggregate PFLDEF structure prefix PFL$;
    BITMAP longword unsigned;                                /*ADDRESS OF START OF BIT MAP
                                                             /*BIT = 1 MEANS AVAILABLE
    STARTBYTE longword unsigned;                             /*STARTING BYTE OFFSET TO SCAN
    SIZE word unsigned;                                      /*SIZE OF PAGE FILE CONTROL BLOCK
    TYPE byte unsigned;                                      /*PAGE FILE CONTROL BLOCK TYPE CODE
    PFC byte unsigned;                                       /*PAGE FAULT CLUSTER FOR PAGE READS
    WINDOW longword unsigned;                                /*WINDOW ADDRESS
    VBN longword unsigned;                                   /*BASE VBN
    BITMAPSIZ longword unsigned;                             /*SIZE IN BYTES OF PAGE FILE
    FREPAGCNT longword unsigned;                             /*COUNT - 1 OF PAGES WHICH MAY BE ALLOCATED
    MAXVBN longword unsigned;                                /*MASK APPLIED TO PTE WITH PAGING FILE
                                                             /* BACKING STORE ADDRESS
    ERRORCNT word unsigned;                                  /*COUNT OF POTENTIALLY BAD PAGES
    ALLOCSIZ byte unsigned;                                  /*CURRENT ALLOCATION REQUEST SIZE
    FLAGS_OVERLAY union fill;
        FLAGS byte unsigned;                                 /*FLAGS BYTE FOR THIS PAGE FILE
        constant "LENGTH" equals . prefix PFL$ tag K;        /*SIZE OF PAGE FILE CONTROL BLOCK
        constant "LENGTH" equals . prefix PFL$ tag C;        /*SIZE OF PAGE FILE CONTROL BLOCK
        FLAGS_BITS structure fill;
            INITED bitfield mask;                            /*THIS PAGE FILE IS USABLE
            PAGFILFUL bitfield mask;                         /*REQUEST FOR PAGING SPACE HAS FAILED
            SWPFILFUL bitfield mask;                         /*REQUEST FOR SWAPPING SPACE HAS FAILED
            CHKPNT bitfield mask;                            /*USEABLE BY CHECKPOINT/RESART
            FILL_1 bitfield length 3 fill prefix PFLDEF tag $$;/*SPARE BITS FOR EXPANSION
            STOPPER bitfield mask;                           /*RESERVED FOR ALL TIME (MUST NEVER BE SET)
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    BITMAPLOC longword unsigned;                             /*BITMAP FOLLOWS PFL HEADER

end PFLDEF;

end_module $PFLDEF;
```

```
module $PFNDEF;

/*+
/* PFN DATA BASE DEFINITIONS
/*-

/*
/* VIELD DEFINITIONS IN PFN$AB_STATE
/*

aggregate PFNDEF  union prefix PFN$;
    PFNDEF_BITS0 structure fill;
        LOC bitfield mask length 3;                         /*LOCATION OF PAGE
/*
/* ***** THE FOLLOWING SPARE BIT MUST BE USED FOR EXTENSION OF THE LOC FIELD
/* ***** OR ALTERNATIVELY THE DELCON BIT MUST BE MOVED ADJACENT TO LOC
/*
        FILL_1 bitfield fill prefix PFNDEF tag $$;          /*NOT IN USE
        DELCON bitfield mask;                               /*DELETE PFN CONTENTS WHEN REF=0
        FILL_2 bitfield length 2 fill prefix PFNDEF tag $$;/*NOT IN USE
        MODIFY bitfield mask;                               /*MODIFY BIT
    end PFNDEF_BITS0;
/*
/* VIELD DEFINITIONS IN PFN$AB_TYPE
/*
    PFNDEF_BITS1 structure fill;
        PAGTYP bitfield mask length 3;                      /*PAGE TYPE
        FILL_3 bitfield fill prefix PFNDEF tag $$;          /*NOT IN USE
        COLLISION bitfield mask;                            /*EMPTY COLLISION QUEUE WHEN PAGE READ COMPLETE
        BADPAG bitfield mask;                               /*BAD PAGE BIT
        RPTEVT bitfield mask;                               /*REPORT EVENT ON I/O COMPLETE
    end PFNDEF_BITS1;
/*
/* VIELD DEFINITIONS IN PFN$AL_BAK
/*
    PFNDEF_BITS2 structure fill;
        BAK bitfield mask length 23;                        /*BACKUP ADDRESS
        GBLBAK bitfield mask;                               /*GLOBAL BACKING STORE ADDRESS
        PGFLX bitfield mask length 8;                       /*PAGE FILE INDEX
    end PFNDEF_BITS2;
/*
/* LOCATION VIELD VALUES
/*
    constant FREPAGLST  equals 0  prefix PFN tag $C;    /*ON FREE PAGE LIST
    constant MFYPAGLST  equals 1  prefix PFN tag $C;    /*ON MODIFIED PAGE LIST
    constant BADPAGLST  equals 2  prefix PFN tag $C;    /*ON BAD PAGE LIST
    constant RELPEND    equals 3  prefix PFN tag $C;    /*RELEASE PENDING
                                                        /*WHEN REFCNT = 0 RELEASE PFN
    constant RDERR      equals 4  prefix PFN tag $C;    /*READ ERROR WHILE PAGING IN
    constant WRTINPROG  equals 5  prefix PFN tag $C;    /*WRITE IN PROGRESS (BY MFY PAG WRITER)
    constant RDINPROG   equals 6  prefix PFN tag $C;    /*READ IN PROGRESS (PAGE IN)
    constant ACTIVE     equals 7  prefix PFN tag $C;    /*PAGE IS ACTIVE AND VALID
/*
/* PAGE TYPE VIELD DEFINITIONS
/*
    constant PROCESS    equals 0  prefix PFN tag $C;    /*PROCESS PAGE
    constant SYSTEM     equals 1  prefix PFN tag $C;     /*SYSTEM PAGE
    constant "GLOBAL"   equals 2  prefix PFN tag $C;     /*GLOBAL PAGE (READ ONLY)
    constant GBLWRT     equals 3  prefix PFN tag $C;     /*GLOBAL WRITABLE PAGE
    constant PPGTBL     equals 4  prefix PFN tag $C;     /*PROCESS PAGE TABLE
    constant GPGTBL     equals 5  prefix PFN tag $C;     /*GLOBAL PAGE TABLE

end PFNDEF;

end_module $PFNDEF;
```

```
module $PHDDEF;
/*+
/* A PROCESS HEADER CONTAINS THE SWAPPABLE SCHEDULER AND
/* MEMORY MANAGEMENT DATA BASES FOR A PROCESS IN THE
/* BALANCE SET.
/*-



aggregate PHDDEF structure prefix PHD$;
    PRIVMSK quadword unsigned;                          /*PRIVILEGE MASK
/*
/* WORKING SET LIST POINTERS - THESE CONTAIN LONG WORD OFFSETS FROM THE
/* BEGINNING OF THE PROCESS HEADER.
/*
    WSLIST word unsigned;                               /*1ST WORKING SET LIST ENTRY
    WSAUTH word unsigned;                               /*AUTHORIZED WORKING SET SIZE
    WSLOCK word unsigned;                               /*1ST LOCKED WORKING SET LIST ENTRY
    WSDYN word unsigned;                                /*1ST DYNAMIC WORKING SET LIST ENTRY
    WSNEXT word unsigned;                               /*LAST WSL ENTRY REPLACED
    WSLAST word unsigned;                               /*LAST WSL ENTRY IN LIST
    WSAUTHEXT word unsigned;                            /*AUTHORIZED WS EXTENT
/*
/* THE FOLLOWING THREE WORDS SPECIFY THE MAXIMUM AND INITIAL WORKING SET
/* SIZES FOR THE PROCESS.  RATHER THAN CONTAINING THE COUNT OF PAGES
/* THEY CONTAIN THE LONG WORD INDEX TO WHAT WOULD BE THE LAST WORKING
/* SET LIST ENTRY.
/*
    WSEXTENT word unsigned;                             /*MAX WORKING SET SIZE AGAINST BORROWING
    WSQUOTA word unsigned;                              /*QUOTA ON WORKING SET SIZE
    DFWSCNT word unsigned;                              /*DEFAULT WORKING SET SIZE
    PAGFIL_OVERLAY union fill;
        PAGFIL longword unsigned;                       /*PAGING FILE INDEX, LONG WORD REF
        PAGFIL_FIELDS structure fill;
            FILL_28 byte dimension 3 fill prefix PHDDEF tag $$;
            PAGFIL byte unsigned;                       /*PAGING FILE INDEX, BYTE REFERENCE
/*
/* PROCESS SECTION TABLE DATA BASE
/*        PSTBASOFF IS THE BYTE OFFSET (INTEGRAL ! OF PAGES) FROM THE
/* BEGINNING OF THE PROCESS HEADER TO THE 1ST LONG WORD BEYOND THE
/* PROCESS SECTION TABLE.
/*        THE WORDS, PSTLAST AND PSTFREE ARE SECTION TABLE INDICES WHICH
/* ARE THE NEGATIVE LONG WORD INDEX FROM THE END OF THE SECTION TABLE TO
/* THE SECTION TABLE ENTRY.
/*
        end PAGFIL_FIELDS;
    end PAGFIL_OVERLAY;
    PSTBASOFF longword unsigned;                        /*BYTE OFFSET TO BASE OF PST
                                                        /*FIRST LONG WORD NOT IN PST
                                                        /*PST GROWS BACKWARDS FROM HERE
    PSTLAST word unsigned;                              /*END OF PROCESS SECTION TABLE
                                                        /*ADR OF LAST PSTE ALLOCATED
    PSTFREE word unsigned;                              /*HEAD OF FREE PSTE LIST
/*
/* CREATE/DELETE PAGE CONTEXT
/*
```

```
    FREP0VA longword unsigned;                          /*1ST FREE VIRTUAL ADR AT END OF P0 SPACE
                                                        /****** MUST BE QUAD WORD AWAY FROM FREP1VA
    FREPTECNT longword unsigned;                        /*CNT OF FREE PTE'S BETWEEN THE ENDS
                                                        /*OF THE P0 AND P1 PAGE TABLES
    FREP1VA longword unsigned;                          /*1ST FREE VIRTUAL ADR AT END OF P1 SPACE
    DFPFC byte unsigned;                                /*DEFAULT PAGE FAULT CLUSTER
    PGTBPFC byte unsigned;                              /*PAGE TABLE CLUSTER FACTOR
    FLAGS_OVERLAY union fill;
        FLAGS word unsigned;                            /*FLAGS WORD
        FLAGS_BITS structure fill;
            PFMFLG bitfield mask;                       /*PAGE FAULT MONITORING ENABLED
            DALCSTX bitfield mask;                      /*NEED TO DEALLOCATE SECTION INDICES
            WSPEAKCHK bitfield mask;                    /*CHECK FOR NEW WORKING SET SIZE (PROC)
            NOACCVIO bitfield mask;                     /*SET AFTER INSWAP OF PROCESS HEADER
            IWSPEAKCK bitfield mask;                    /*CHECK FOR NEW WORKING SET SIZE (IMAGE)
            IMGDMP bitfield mask;                       /*TAKE IMAGE DUMP ON ERROR EXIT
            NO_WS_CHNG bitfield mask;                   /*NO CHANGE TO SORKING SET OR SWAPPING
                                                        /* SHORT TERM USE BY MMG CODE ONLY
        end FLAGS_BITS;
/*
/* QUOTAS AND LIMITS
/*
    end FLAGS_OVERLAY;
    CPUTIM longword unsigned;                           /*ACCUMULATED CPU TIME CHARGED
    QUANT word unsigned;                                /*ACCUMULATED CPU TIME SINCE
                                                        /*LAST QUANTUM OVERFLOW
    PRCLM word unsigned;                                /*SUBPROCESS QUOTA
    ASTLM word unsigned;                                /*AST LIMIT

    PHVINDEX word unsigned;                             /*PROCESS HEADER VECTOR INDEX
    BAK longword unsigned;                              /*POINTER TO BACKUP ADDRESS VECTOR FOR
                                                        /*PROCESS HEADER PAGES
    WSLX_OVERLAY union fill;
        WSLX longword unsigned;                         /*POINTER TO WORKING SET LIST INDEX
                                                        /*SAVE AREA
        PSTBASMAX longword unsigned;                    /*LW OFFSET TO TOP PST ADDRESS
    end WSLX_OVERLAY;
    PAGEFLTS longword unsigned;                         /*COUNT OF PAGE FAULTS
    WSSIZE word unsigned;                               /*CURRENT ALLOWED WORKING SET SIZE
    SWAPSIZE word unsigned;                             /*CURRENT SWAP BLOCK ALLOCATION

/*
/* THE NEXT TWO I/O COUNTERS MUST BE ADJACENT
/*

    DIOCNT longword unsigned;                           /*DIRECT I/O COUNT
    BIOCNT longword unsigned;                           /*BUFFERED I/O COUNT

    CPULIM longword unsigned;                           /*LIMIT ON CPU TIME FOR PROCESS
    CPUMODE byte unsigned;                              /*ACCESS MODE TO NOTIFY ABOUT CPUTIME
    AWSMODE byte unsigned;                              /*ACCESS MODE FLAG FOR AUTO WS AST
    FILL_30 word unsigned;                              /*SPARE
/*
/* PAGE TABLE STATISTICS
/*
    PTWSLELCK longword unsigned;                        /* BYTE OFFSET TO BYTE ARRAY OF COUNTS
```

```
                                                            /* OF LOCKED WSLE'S IN THIS PAGE TABLE
    PTWSLEVAL longword unsigned;                            /* BYTE OFFSET TO BYTE ARRAY OF COUNTS
                                                            /* OF VALID WSLE'S IN THIS PAGE TABLE
    constant PHDPAGCTX  equals 8  prefix PHD tag $C;        /* SIZE OF CONTEXT FOR PHD PAGES
    PTCNTLCK word unsigned;                                 /* COUNT OF PAGE TABLES CONTAINING
                                                            /* 1 OR MORE LOCKED WSLE
    PTCNTVAL word unsigned;                                 /* COUNT OF PAGE TABLES CONTAINING
                                                            /* 1 OR MORE VALID WSLE
    PTCNTACT word unsigned;                                 /* COUNT OF ACTIVE PAGE TABLES
    PTCNTMAX word unsigned;                                 /* MAX COUNT OF PAGE TABLES
                                                            /* WHICH HAVE NON-ZERO PTE'S
    WSFLUID word unsigned;                                  /* GUARANTEED NUMBER OF FLUID WS PAGES
    EXTDYNWS word unsigned;                                 /* EXTRA DYNAMIC WORKING SET LIST ENTRIES
                                                            /* ABOVE REQUIRED WSFLUID MINIMUM
/*
/* HARDWARE PCB PORTION OF PROCESS HEADER
/*
    PCB_OVERLAY union fill;
        PCB longword unsigned;                              /*HARDWARE PCB
        KSP longword unsigned;                              /*KERNEL STACK POINTER
    end PCB_OVERLAY;
    ESP longword unsigned;                                  /*EXEC STACK POINTER
    SSP longword unsigned;                                  /*SUPERVISOR STACK POINTER
    USP longword unsigned;                                  /*USER STACK POINTER
    R0 longword unsigned;                                   /*R0
    R1 longword unsigned;                                   /*R1
    R2 longword unsigned;                                   /*R2
    R3 longword unsigned;                                   /*R3
    R4 longword unsigned;                                   /*R4
    R5 longword unsigned;                                   /*R5
    R6 longword unsigned;                                   /*R6
    R7 longword unsigned;                                   /*R7
    R8 longword unsigned;                                   /*R8
    R9 longword unsigned;                                   /*R9
    R10 longword unsigned;                                  /*R10
    R11 longword unsigned;                                  /*R11
    R12 longword unsigned;                                  /*R12
    R13 longword unsigned;                                  /*R13
    PC longword unsigned;                                   /*PC
    PSL longword unsigned;                                  /*PROGRAM STATUS LONGWORD
    P0BR longword unsigned;                                 /*P0 BASE REGISTER
    P0LRASTL_OVERLAY union fill;
        P0LRASTL longword unsigned;                         /*P0LR, ASTLVL
        P0LRASTL_BITS structure fill;
            P0LR bitfield length 24;                        /*P0 LENGTH REGISTER
            ASTLVL bitfield length 8;                       /* AST LEVEL
        end P0LRASTL_BITS;
        P0LRASTL_FIELDS structure fill;
            FILL_29 byte dimension 3 fill prefix PHDDEF tag $$;
            ASTLVL byte unsigned;                           /*AST LEVEL SUBFIELD
        end P0LRASTL_FIELDS;
    end P0LRASTL_OVERLAY;
    P1BR longword unsigned;                                 /*P1 BASE REGISTER
    P1LR longword unsigned;                                 /*P1 LENGTH REGISTER
    EMPTPG word unsigned;                                   /*COUNT OF EMPTY WORKING SET PAGES
    RESPGCNT word unsigned;                                 /*RESIDENT PAGE COUNT
    REQPGCNT word unsigned;                                 /*REQUIRED PAGE COUNT
    CWSLX word unsigned;                                    /*CONTINUATION WSLX
    AUTHPRIV quadword unsigned;                             /*AUTHORIZED PRIVILEGES MASK
    IMAGPRIV quadword unsigned;                             /*INSTALLED IMAGE PRIVILEGES MASK
    RESLSTH longword unsigned;                              /*POINTER TO RESOURCE LIST
    IMGCNT longword unsigned;                               /*IMAGE COUNTER BUMPED BY SYSRUNDWN
    PFLTRATE longword unsigned;                             /*PAGE FAULT RATE
    PFLREF longword unsigned;                               /*PAGE FAULTS AT END OF LAST INTERVAL
    TIMREF longword unsigned;                               /*TIME AT END OF LAST INTERVAL
    MPINHIBIT longword unsigned;                            /*COUNT OF REASONS WHY PROCESS
                                                            /* MUST RUN ON PRIMARY IN MP SYSTEM
    PGFLTIO longword unsigned;                              /*COUNT OF PAGEFAULT I/O
    AUTHPRI byte unsigned;                                  /*INITIAL PROCESS PRIORITY
    FILL_1 byte fill prefix PHDDEF tag $$;                  /*SPARE
    FILL_2 word fill prefix PHDDEF tag $$;                  /*SPARE
    EXTRACPU longword unsigned;                             /*ACCUMULATED CPU TIME LIMIT EXTENSION
    MIN_CLASS structure;                     /* MINIMUM AUTHORIZED SECURITY CLEARANCE
        FILL_3 byte unsigned dimension 20 fill tag $$;
        end MIN_CLASS;
    MAX_CLASS structure;                     /* MAXIMUM AUTHORIZED SECURITY CLEARANCE
        FILL_4 byte unsigned dimension 20 fill tag $$;
        end MAX_CLASS;
    SPARE longword unsigned;                                /*SPARE
    FILL_13 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_14 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_15 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_16 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_17 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_18 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_19 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_20 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_21 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_22 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_23 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_24 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_25 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_26 longword fill prefix PHDDEF tag $$;             /*SPARE
    FILL_27 longword fill prefix PHDDEF tag $$;             /*SPARE
/*
/* END OF FIXED PORTION OF PROCESS HEADER
/*
    constant "LENGTH" equals . prefix PHD$ tag K;           /*LENGTH OF FIXED PART OF PROCESS HEADER
    constant "LENGTH" equals . prefix PHD$ tag C;           /*LENGTH OF FIXED PART OF PROCESS HEADER
    WSL longword unsigned;                                  /*FIRST WORKING SET LIST ENTRY

end PHDDEF;

end_module $PHDDEF;
```

```
module $PIBDEF;
/*+
/* PERFORMANCE I/O INFORMATION BLOCK
/*-


aggregate PIBDEF union prefix PIB$;
    TYPE byte unsigned;                                  /*TYPE OF ENTRY

/*
/* START OF I/O REQUEST TRANSACTION MESSAGE BLOCK
/*

end PIBDEF;

aggregate PIBDEF1 structure prefix PIB$;
    FILL_5 byte fill prefix PIBDEF tag $$;
    SRQ_PRI byte unsigned;                               /*BASE PRIORITY OF PROCESS
    SRQ_ACON word unsigned;                              /*Access control info from WCB or 0
    SRQ_TIME quadword unsigned;                          /*TIME OF I/O TRANSACTION
    SRQ_SEQN longword unsigned;                          /*SEQUENCE NUMBER OF I/O TRANSACTION
    SRQ_PID longword unsigned;                           /*REQUESTER PID
    SRQ_UCB longword unsigned;                           /*ADDRESS OF DEVICE UCB
    SRQ_FUNC word unsigned;                              /*I/O FUNCTION CODE
    SRQ_STS word unsigned;                               /*I/O PACKET STATUS
    SRQ_ACCESS byte unsigned;                            /*Access control info from WCB or 0
    FILL_1 byte dimension 3 fill prefix PIBDEF tag $$;   /*SPARE UNUSED BYTES
    constant SRQ_SIZE equals . prefix PIB$ tag K;        /*LENGTH OF START I/O MESSAGE
    constant SRQ_SIZE equals . prefix PIB$ tag C;        /*LENGTH OF START I/O MESSAGE

/*
/* START OF I/O TRANSACTION MESSAGE BLOCK
/*

end PIBDEF1;

aggregate PIBDEF2 structure prefix PIB$;
    FILL_6 byte fill prefix PIBDEF tag $$;
    FILL_2 byte fill prefix PIBDEF tag $$;               /*SPARE UNUSED BYTE
    FILL_9 word fill prefix PIBDEF tag $$;               /*SPARE UNUSED WORD
    SIO_TIME quadword unsigned;                          /*TIME OF TRANSACTION
    SIO_SEQN longword unsigned;                          /*SEQUENCE NUMBER OF TRANSACTION
    SIO_MEDIA longword unsigned;                         /*TRANSFER MEDIA ADDRESS
    SIO_BCNT longword unsigned;                          /*TRANSFER BYTE COUNT
    constant SIO_SIZE equals . prefix PIB$ tag K;        /*LENGTH OF I/O TRANSACTION MESSAGE
    constant SIO_SIZE equals . prefix PIB$ tag C;        /*LENGTH OF I/O TRANSACTION MESSAGE

/*
/* END OF I/O TRANSACTION MESSAGE BLOCK
/*

end PIBDEF2;

aggregate PIBDEF3 structure prefix PIB$;
    FILL_7 byte fill prefix PIBDEF tag $$;
    FILL_3 byte dimension 3 fill prefix PIBDEF tag $$;  /*SPARE UNUSED BYTES
    EIO_TIME quadword unsigned;                         /*TIME OF TRANSACTION
    EIO_SEQN longword unsigned;                         /*SEQUENCE NUMBER OF TRANSACTION
    EIO_IOSB quadword unsigned;                         /*FINAL I/O STATUS
    constant EIO_SIZE equals . prefix PIB$ tag K;       /*LENGTH OF END OF I/O TRANSACTION
    constant EIO_SIZE equals . prefix PIB$ tag C;       /*LENGTH OF END OF I/O TRANSACTION

/*
/* END OF I/O REQUEST MESSAGE BLOCK
/*

end PIBDEF3;

aggregate PIBDEF4 structure prefix PIB$;
    FILL_8 byte fill prefix PIBDEF tag $$;
    FILL_4 byte dimension 3 fill prefix PIBDEF tag $$;  /*SPARE UNUSED BYTES
    ERQ_TIME quadword unsigned;                         /*TIME OF TRANSACTION
    ERQ_SEQN longword unsigned;                         /*SEQUENCE NUMBER OF TRANSACTION
    constant ERQ_SIZE equals . prefix PIB$ tag K;       /*LENGTH OF END OF I/O REQUEST TRANSACTION
    constant ERQ_SIZE equals . prefix PIB$ tag C;       /*LENGTH OF END OF I/O REQUEST TRANSACTION

/*
/* I/O MESSAGE BLOCK ENTRY TYPE CODES
/*

    constant SRQ        equals 0  prefix PIB tag $K;    /*START OF I/O REQUEST
    constant SIO        equals 1  prefix PIB tag $K;    /*START OF I/O TRANSACTION
    constant EIO        equals 2  prefix PIB tag $K;    /*END OF I/O TRANSACTION
    constant ERQ        equals 3  prefix PIB tag $K;    /*END OF I/O REQUEST
    constant ARQ        equals 4  prefix PIB tag $K;    /*ABORTED I/O REQUEST

end PIBDEF4;

aggregate PIBDEF5 structure prefix PIB$;
    FILL_10 byte fill prefix PIBDEF tag $$;
    FILL_11 byte dimension 3 fill prefix PIBDEF tag $$; /*SPARE UNUSED BYTES
    ARQ_TIME quadword unsigned;                         /*TIME OF TRANSACTION
    ARQ_SEQN longword unsigned;                         /*SEQUENCE NUMBER OF TRANSACTION
    constant ARQ_SIZE equals . prefix PIB$ tag K;       /*LENGTH OF ABORTED I/O TRANSACTION
    constant ARQ_SIZE equals . prefix PIB$ tag C;       /*LENGTH OF ABORTED I/O TRANSACTION

/*
/* ABORTED I/O REQUEST MESSAGE BLOCK
/*

end PIBDEF5;

end_module $PIBDEF;
```

```
                                                       6 8
SYSDEFMP.SDL;1                                16-SEP-1984 16:45:31.57 Page  69

module $PMBDEF;
/*+
/* PAGE FAULT MONITOR CONTROL BLOCK
/*-



aggregate PMBDEF structure prefix PMB$;
    CURBUF longword unsigned;                           /*Current buffer pointer
    BUFBASE longword unsigned;                          /*Current buffer base address
    SIZE word unsigned;                                 /*Block size field
    TYPE byte unsigned;                                 /*Dynamic structure type (PMB)
    FLAGS structure byte unsigned;                      /*Processing flags
        MODE bitfield mask;                             /*Mode of operation
            constant "SUBPROC" equals 0; /*Subprocess mode
            constant "IMAGE"   equals 1; /*Image mode
        ASTIP bitfield mask;                            /*AST in progress flag
        QAST bitfield mask;                             /*Imbedded ACB is enqueued on the PCB
    end FLAGS;
    LASTCPU longword unsigned;                          /*Last recorded CPU time
    OVERFLOW longword unsigned;                         /*Buffer overflow counter (both modes)
    HDR quadword unsigned;                              /*Free   buffer queue header
    SBPHDR quadword unsigned;                           /*Filled buffer queue header
    ACB_OVERLAY union fill;
        AST_BLOCK structure fill;                       /*Used as AST block in image mode
            ASTQFL longword unsigned;                   /*ACB flink
            ASTQBL longword unsigned;                   /*ACB blink
            SPARE_1 byte dimension 2 fill prefix PMBDEF tag $$; /*SPARE
            ACMODE byte unsigned;                       /*Owner access mode
            RMOD byte unsigned;                         /*AST delivery mode/flags
            PID longword unsigned;                      /*PID for AST delivery
            AST longword unsigned;                      /*AST routine address
            ASTPRM longword unsigned;                   /*AST parameter
            KAST longword unsigned;                     /*Address of piggy-back kernel AST routine
        end AST_BLOCK;
        SUBP_BLOCK structure fill;                      /*Utility storage in subprocess mode
            SPARE_2 longword dimension 2 fill prefix PMBDEF tag $$; /*SPARE
            MBXCHN word unsigned;                       /*Subprocess mailbox channel
            OACMODE byte unsigned;                      /*Owner access mode (Synonym for ACMODE)
            SPARE_3 byte dimension 1 fill prefix PMBDEF tag $$; /*SPARE
            IPID longword unsigned;                     /*IPID of subprocess (Synonym for PID)
            EPID longword unsigned;                     /*EPID of subprocess
            SPARE_4 longword dimension 2 fill prefix PMBDEF tag $$; /*SPARE
        end SUBP_BLOCK;
    end ACB_OVERLAY;
    constant "LENGTH" equals . prefix PMB$ tag K;       /*Length of PMB
    constant "LENGTH" equals . prefix PMB$ tag C;       /*Length of PMB
end PMBDEF;

end_module $PMBDEF;
```

```
module $PQBDEF;
/*+
/* PROCESS QUOTA BLOCK DEFINITION
/*-

aggregate PQBDEF structure prefix PQB$;
    PRVMSK quadword unsigned;                       /* PRIVILEGE MASK
    SIZE word unsigned;                             /* SIZE OF PQB IN BYTES
    TYPE byte unsigned;                             /* STRUCTURE TYPE CODE
    STS byte unsigned;                              /* STATUS FLAGS

    ASTLM longword unsigned;                        /* AST LIMIT
    BIOLM longword unsigned;                        /* BUFFERED I/O LIMIT
    BYTLM longword unsigned;                        /* BUFFERED I/O LIMIT
    CPULM longword unsigned;                        /* CPU TIME LIMIT
    DIOLM longword unsigned;                        /* DIRECT I/O LIMIT
    FILLM longword unsigned;                        /* OPEN FILE LIMIT
    PGFLQUOTA longword unsigned;                    /* PAGING FILE QUOTA
    PRCLM longword unsigned;                        /* SUB-PROCESS LIMIT
    TQELM longword unsigned;                        /* TIMER QUEUE ENTRY LIMIT
    WSQUOTA longword unsigned;                      /* WORKING SET QUOTA
    WSDEFAULT longword unsigned;                    /* WORKING SET DEFAULT
    ENQLM longword unsigned;                        /* ENQUEUE LIMIT
    WSEXTENT longword unsigned;                     /* MAXIMUM WORKING SET SIZE
    JTQUOTA longword unsigned;                      /* JOB-WIDE LOGICAL NAME TABLE CREATION QUOTA

    FLAGS structure word unsigned;                  /* MISC FLAGS
        IMGDMP bitfield mask;                       /* TAKE IMAGE DUMP ON SERIOUS ERROR
        end FLAGS;
    MSGMASK byte unsigned;                          /* MESSAGE FLAGS
    FILL_1 byte unsigned;                           /* Spare
    UAF_FLAGS longword unsigned;                    /* FLAGS FROM UAF RECORD
    CREPRC_FLAGS longword unsigned;                 /* FLAGS FROM $CREPRC ARGUMENT LIST

    MIN_CLASS structure;                            /* MINIMUM AUTHORIZED SECURITY CLEARANCE
        FILL_2 byte unsigned dimension 20 fill tag $$;
        end MIN_CLASS;

    MAX_CLASS structure;                            /* MAXIMUM AUTHORIZED SECURITY CLEARANCE
        FILL_3 byte unsigned dimension 20 fill tag $$;
        end MAX_CLASS;

    INPUT_ATT longword unsigned;                    /*  SYS$INPUT attributes
    OUTPUT_ATT longword unsigned;                   /*  SYS$OUTPUT attributes
    ERROR_ATT longword unsigned;                    /*  SYS$ERROR attributes
    DISK_ATT longword unsigned;                     /*  SYS$DISK attributes

    CLI_NAME character length 32;                   /* CLI name
    CLI_TABLE character length 256;                 /* CLI table name
    SPAWN_CLI character length 32;                  /* Spawn CLI name
    SPAWN_TABLE character length 256;               /* Spawn CLI table name

    INPUT character length 256;                     /* LOGICAL NAME FOR INPUT
    OUTPUT character length 256;                    /* LOGICAL NAME FOR OUTPUT
    ERROR character length 256;                     /* LOGICAL NAME FOR ERROR OUTPUT
    DISK character length 256;                  /* LOGICAL NAME FOR SYS$DISK

    DDSTRING character length 256;              /* DEFAULT DIRECTORY STRING

    IMAGE character length 256;                 /* IMAGE NAME FOR NEW PROCESS

    constant 'LENGTH' equals . prefix PQB$ tag K;       /* LENGTH OF PROCESS QUOTA BLOCK
    constant 'LENGTH' equals . prefix PQB$ tag C;       /* LENGTH OF PROCESS QUOTA BLOCK

end PQBDEF;

end_module $PQBDEF;
```

```
module $PRBDEF;

/*+
/*
/* Protection block definition. The protection block is used to specify
/* protection on objects internal to the system (e.g., devices, logical
/* name tables, etc.) It is used as input to the EXE$CHECKACCESS routine.
/*
/*-

aggregate PRBDEF structure prefix PRB$;
    FLAGS structure word unsigned;      /* Presence flag bits
        UIC bitfield mask;              /* Set for simple UIC protection
        ACL bitfield mask;              /* Set for access control list
        CLASS bitfield mask;            /* Set for security classification
        CLASSMAX bitfield mask;         /* Set for security class range
    end FLAGS;
    PROTECTION word unsigned;           /* SOGW protection mask
    OWNER longword unsigned;            /* Owner UIC

/*
/* The remaining items in the protection block are optional and therefore
/* do not have fixed offsets. The description given below is for a
/* hypothetical fully configured protection block.
/*
/*  ACL quadword;                       /* ACL listhead
/*  CLASS structure;                    /* Classification mask
/*      FILL_1 long dimension 5 fill;
/*      end CLASS;
/*  CLASSMAX structure;                 /* Maximum class mask for range
/*      FILL_2 long dimension 5 fill;
/*      end CLASSMAX;
/*

end PRBDEF;
end_module $PRBDEF;
```

```
module $PRCPOLDEF;
/*+
/* PROCESS POLLER MAILBOX MESSAGE DEFINITIONS
/*-

aggregate PRCPOLDEF structure prefix PRCPOL$;
    SYSIDL longword unsigned;                              /*LOW ORDER SYSTEM ID
    SYSIDH word unsigned;                                  /*HIGH ORDER SYSTEM ID
    FILL_1 word unsigned fill;                             /* (UNUSED)
    NODNAM character length 16;                            /*SCA NODE NAME (COUNTED ASCII)
    PRCNAM byte unsigned dimension 16;                     /*PROCESS NAME
    DIRINF byte unsigned dimension 16;                     /*DIRECTORY INFORMATION
    constant "SIZ" equals . prefix PRCPOL$ tag C;          /*SIZE OF MESSAGE

end PRCPOLDEF;

end_module $PRCPOLDEF;
```

```
module $PRIDEF;

/*+
/* PRIORITY INCREMENT CLASS DEFINITIONS
/*-

constant NULL   equals 0  prefix PRI tag $;            /* NO PRIORITY INCREMENT
constant IOCOM  equals 1  prefix PRI tag $;            /* DIRECT I/O COMPLETION
constant RESAVL equals 2  prefix PRI tag $;            /* RESOURCE AVAIL
constant TOCOM  equals 3  prefix PRI tag $;            /* TERMINAL OUTPUT COMPLETE
constant TICOM  equals 4  prefix PRI tag $;            /* TERMINAL INPUT COMPLETE
constant TIMER  equals 2  prefix PRI tag $;            /* TIMER INTERVAL COMPLETION

end_module $PRIDEF;
```

```
module $PRMDEF;
/*+
/* DEFINE PARAMETER DESCRIPTOR BLOCK
/*-


aggregate PRM_DEF structure prefix PRM$;
    ADDR longword unsigned;                                 /*ADDRESS OF PARAMETER
    "DEFAULT" longword unsigned;                            /*DEFAULT VALUE
    MIN longword unsigned;                                  /*MINIMUM VALUE (-1)=>NONE
    MAX longword unsigned;                                  /*MAXIMUM VALUE (-1)=>NONE
    FLAGS_OVERLAY union fill;
        FLAGS longword unsigned;                            /*TYPE FLAGS
        FLAGS_BITS structure fill;
            DYNAMIC bitfield mask;                          /* DYNAMIC PARAMETER
            STATIC bitfield mask;                           /* STATIC PARAMETER
            SYSGEN bitfield mask;                           /* SYSGEN PARAMETER
            ACP bitfield mask;                              /* ACP CONTROL PARAMETER
            JBC bitfield mask;                              /* JOB CONTROL PARAMETER
            RMS bitfield mask;                              /* RMS CONTROL PARAMETER
            SYS bitfield mask;                              /* GENERAL SYSTEM PARAMETER
            SPECIAL bitfield mask;                          /* SPECIAL PARAMETER
            DISPLAY bitfield mask;                          /* DISPLAY ONLY (NO CHANGE)
            CONTROL bitfield mask;                          /* CONTROL PARAMETER
            MAJOR bitfield mask;                            /* MAJOR PARAMETER
            PQL bitfield mask;                              /* PROCESS QUOTA LIST
            NEG bitfield mask;                              /* NEGATIVE
            TTY bitfield mask;                              /* TERMINAL CONTROL PARAMETER
            SCS bitfield mask;                              /* SCS CONTROL PARAMETERS
            CLUSTER bitfield mask;                          /* CLUSTER CONTROL PARAMETERS
            ASCII bitfield mask;                            /* ASCII PARAMETER
            LGI bitfield mask;                              /* LOGIN PARAMETER
        end FLAGS_BITS;
    end FLAGS_OVERLAY;
    SIZE byte unsigned;                                     /*SIZE CODE FOR DATUM
    constant "BYTE"      equals 8   prefix PRM tag $C;      /*
    constant "WORD"      equals 16  prefix PRM tag $C;      /*
    constant LONG        equals 32  prefix PRM tag $C;      /*
    constant "QUAD"      equals 64  prefix PRM tag $C;      /*
    constant "OCTA"      equals 128 prefix PRM tag $C;      /*
    POS byte unsigned;                                      /*BIT POSITION
    NAME character length 16;                               /*ASCIC NAME STRING
    constant MAXNAMLEN  equals 15  prefix PRM tag $C;       /*MAXIMUM LENGTH FOR PARAMETER NAME
    UNIT character length 12;                               /*ASCIC UNIT STRING
    constant MAXUNILEN  equals 11  prefix PRM tag $C;       /*MAXIMUM LENGTH FOR UNIT NAME
    constant "LENGTH" equals . prefix PRM$ tag K;           /*SIZE OF DESCRIPTOR BLOCK
    constant "LENGTH" equals . prefix PRM$ tag C;           /*SIZE OF DESCRIPTOR BLOCK
end PRM_DEF;

end_module $PRMDEF;
```

```
module $PRQDEF;
/*+
/* INTER-PROCESSOR REQUEST BLOCK DEFINITIONS
/*
/* THIS IS THE BASIC FORMAT FOR AN EXECUTIVE OR DRIVER REQUEST FROM
/* ONE PROCESSOR TO ANOTHER PROCESSOR.
/*-


aggregate PRQDEF structure prefix PRQ$;
    FLINK longword unsigned;                                /*FORWARD LINK TO NEXT BLOCK
    BLINK longword unsigned;                                /*BACKWARD LINK TO PREVIOUS BLOCK
    FILL_1 longword dimension 4 fill prefix PRQDEF tag $$;/* (RESERVED FOR FORK CONTEXT)
    TO_PORT word unsigned;                                  /*PORT NUMBER TO SEND REQUEST TO
    FR_PORT word unsigned;                                  /*PORT NUMBER REQUEST IS FROM
    DISPATCH word unsigned;                                 /*MESSAGE DISPATCHER ID
                                                            /* MESSAGE DISPATCHER ID'S
    constant EXEC       equals 0  prefix PRQ tag $C;        /* EXECUTIVE REQUEST ID
    constant MAILBOX    equals 1  prefix PRQ tag $C;        /* MAILBOX REQUEST ID
    constant REMDISK    equals 2  prefix PRQ tag $C;        /* REMOTE DISK REQUEST ID
    constant HSC50      equals 3  prefix PRQ tag $C;        /* HSC-50 REQUEST ID
    FILL_2 word fill prefix PRQDEF tag $$;                  /*(UNUSED)
/*
    REQTYPE word unsigned;                                  /*REQUEST TYPE
                                                            /* MESSAGE DISPATCHER REQUEST SUB-TYPES
    constant SETEF      equals 0  prefix PRQ tag $C;        /* COPY COMMON EVENT FLAG REQUEST ID
    constant RESAVL     equals 1  prefix PRQ tag $C;        /* REPORT RESOURCE AVAILABLE
    UNIT word unsigned;                                     /*UNIT NUMBER
    PARAM longword unsigned;                                /*FIRST PARAMETER
    constant MINLENGTH  equals 64  prefix PRQ tag $C;       /*MINIMUM REQUEST BLOCK LENGTH

end PRQDEF;

end_module $PRQDEF;
```

```
module $PSMDEF;                             /* Print symbiont definitions

/*+
/* Symbolic definitions for print symbionts.
/*
/*      Public definition of various constants and data structures
/*      used by the standard VMS print symbiont, and by user modified
/*      print symbionts.
/*
/*-


/*
/*      Service routine function codes
/*

constant (

  /*
  /* IO functions
  /*

    CANCEL,                         /* Cancel pending operations
    CLOSE,                          /* Release resources
    FORMAT,                         /* Format buffer
    OPEN,                           /* Obtain resources
    READ,                           /* Read
    GET_KEY,                        /* Read record key
    POSITION_TO_KEY,                /* Read by record context
    REWIND,                         /* Rewind file
    WRITE,                          /* Write
    WRITE_NOFORMAT,                 /* Write with driver formatting disabled
    WRITE_SUPPRESSED,               /* Write but suppress output

  /*
  /* Message notification functions
  /*

    PAUSE_TASK,                     /* STOP /QUEUE
    RESET_STREAM,                   /* STOP /QUEUE /RESET
    RESUME_TASK,                    /* START /QUEUE (when paused)
    START_STREAM,                   /* START /QUEUE (when stopped)
    START_TASK,                     /* (ofiginated by job controller)
    STOP_TASK,                      /* STOP /QUEUE /ABORT or /REQUEUE
    STOP_STREAM                     /* STOP /QUEUE /NEXT

) equals 1 increment 1 prefix PSM$;


/*
/*      Replacement routines
/*

constant (
 /*
 /* Task services -- where applicable the ordering of these literals
 /* determines the sequence of the corresponding service routines.
 /*

 /*
 /* Page services
 /*

   PAGE_SETUP,                  /* Page setup        - page setup modules
   PAGE_HEADER,                 /* Page separation   - page headers


 /*
 /* Library module service
 /*

   LIBRARY_INPUT,               /* Module services


 /*
 /* Filter services
 /*

   INPUT_FILTER,                /* Filter service    - input
   MAIN_FORMAT,                 /* Format service    - carriage control
   OUTPUT_FILTER,               /* Filter service    - output


 /*
 /* Output services
 /*

   OUTPUT,                      /* Main output routine


 /*
 /* General input services
 /*

   JOB_SETUP,                   /* Job setup         - job reset modules
   FORM_SETUP,                  /* Form setup        - form setup modules
   JOB_FLAG,                    /* Job separation    - flag page
   JOB_BURST,                   /* Job separation    - burst page
   FILE_SETUP,                  /* File setup        - file setup modules
   FILE_FLAG,                   /* File separation   - flag page
   FILE_BURST,                  /* File separation   - burst page
   FILE_SETUP_2,                /* File setup        - top of form
   MAIN_INPUT,                  /* File service      - main routine
   FILE_INFORMATION,            /* Additional information print
   FILE_ERRORS,                 /* Errors during task processing
   FILE_TRAILER,                /* File separation   - trailer page
   JOB_RESET,                   /* Job reset         - job reset modules
   JOB_TRAILER,                 /* Job separation    - trailer page
   JOB_COMPLETION,              /* Job completion    - top of form
```

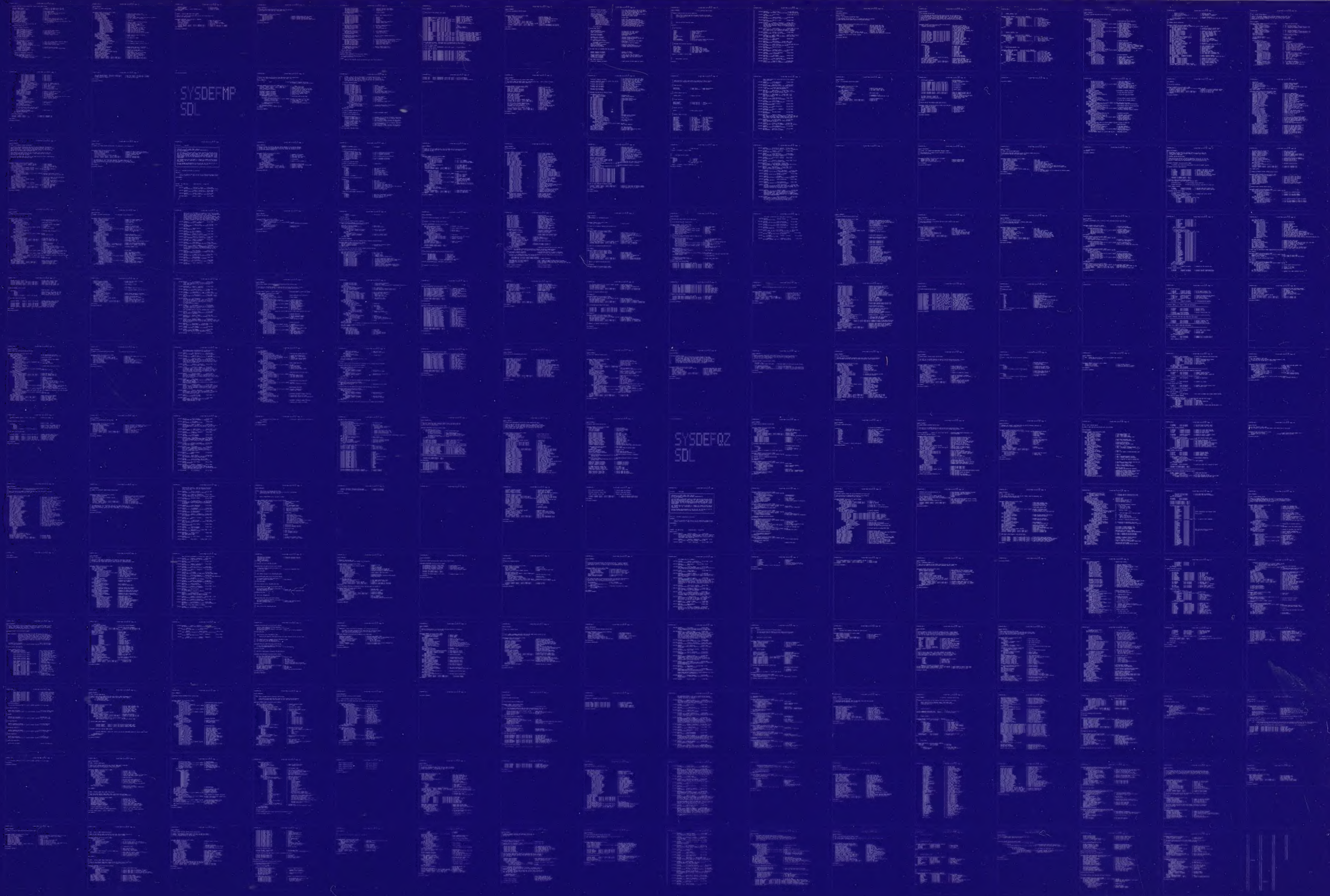
```
    max                          /* MUST BE LAST

) equals 1 increment 1 prefix PSM$;


/*
/*      Carriage control types
/*

constant (
    INTERNAL,                    /*  - imbedded
    IMPLIED,                     /*  - implied
    FORTRAN,                     /*  - fortran
    PRINT,                       /*  - print file (PRN)
    MAX                          /* MUST BE LAST

) equals 1 increment 1 prefix PSM$ tag K_CC;

end_module $PSMDEF;
```

```
module $PTEDEF;

/*+
/* DEFINE PAGE TABLE ENTRY VIELDS AND VALUES
/*-
/*
/* VIELD DEFINITION FOR "VALID" PTE'S
/*

aggregate PTEDEF  union prefix PTE$;
    PTEDEF_BITS0 structure fill;
        PFN bitfield mask length 21;                    /* PAGE FRAME NUMBER
        WINDOW bitfield mask;                           /* WINDOW BIT
        FILL_1 bitfield fill prefix PTEDEF tag $$;      /* RESERVED
        OWN bitfield mask length 2;                     /* MODE OF THE OWNER
        FILL_2 bitfield fill prefix PTEDEF tag $$;      /* RESERVED
        MODIFY bitfield mask;                           /* MODIFY BIT
        PROT bitfield mask length 4;                    /* FROTECTION
        VALID bitfield mask;                            /* VALID BIT
    end PTEDEF_BITS0;
/*
/* VIELD DEFINITIONS FOR VARIOUS INVALID FORMS OF PTE
/*
    PTEDEF_BITS1 structure fill;
        STX bitfield length 16 signed;                          /* SECTION TABLE INDEX
        CRF bitfield mask;                              /* COPY ON REFERENCE
        DZRO bitfield mask;                             /* DEMAND ZERO
        WRT bitfield mask;                              /* SECTION FILE IS ACCESSED FOR WRITING
        FILL_3 bitfield length 3 fill prefix PTEDEF tag $$;/* SPARE
        TYP0 bitfield mask;                             /* LOW ORDER BIT OF PTE TYPE
        FILL_4 bitfield length 2 fill prefix PTEDEF tag $$;/* OWNER FIELD
        FILL_5 bitfield fill prefix PTEDEF tag $$;      /* RESERVED
        TYP1 bitfield mask;                             /* HIGH ORDER BIT OF PTE TYPE
                                                        /* OVERLAYS MODIFY BIT
    end PTEDEF_BITS1;
    PTEDEF_BITS2 structure fill;
        PGFLVB bitfield mask length 22;                 /* PAGE FILE VBN
    end PTEDEF_BITS2;
    PTEDEF_BITS3 structure fill;
        FILL_6 bitfield length 21 fill prefix PTEDEF tag $$;/* SPACING
        CHKPNT bitfield mask;                           /* FORGET THAT THIS PAGE HAS A BACKING STORE
    end PTEDEF_BITS3;
                                                        /*  TO BE FORGOTTEN

    PTEDEF_BITS4 structure fill;
        GPTX bitfield mask length 22;                   /* GLOBAL PAGE TABLE INDEX
    end PTEDEF_BITS4;
/*+
/* PROTECTION FIELD DEFINITIONS
/*-
    constant NA    equals 0 prefix PTE tag $C;          /* NO ACCESS
    constant KR    equals %X18000000 prefix PTE tag $C;  /* KERNEL READ ONLY
    constant KW    equals %X10000000 prefix PTE tag $C;  /* KERNEL WRITE
    constant ER    equals %X38000000 prefix PTE tag $C;  /* EXEC READ ONLY
    constant EW    equals %X28000000 prefix PTE tag $C;  /* EXEC WRITE
    constant SR    equals %X58000000 prefix PTE tag $C;  /* SUPER READ ONLY
    constant SW   equals %X40000000 prefix PTE tag $C;  /* SUPER WRITE
    constant UR   equals %X78000000 prefix PTE tag $C;  /* USER READ ONLY
    constant UW   equals %X20000000 prefix PTE tag $C;  /* USER WRITE
    constant ERKW equals %X30000000 prefix PTE tag $C;  /* EXEC READ KERNEL WRITE
    constant SRKW equals %X50000000 prefix PTE tag $C;  /* SUPER READ KERNEL WRITE
    constant SREW equals %X48000000 prefix PTE tag $C;  /* SUPER READ EXEC WRITE
    constant URKW equals %X70000000 prefix PTE tag $C;  /* USER READ KERNEL WRITE
    constant UREW equals %X68000000 prefix PTE tag $C;  /* USER READ EXEC WRITE
    constant URSW equals %X60000000 prefix PTE tag $C;  /* USER READ SUPER WRITE
/*+
/* OWNER FIELD DEFINITIONS
/*-
    constant KOWN equals 0 prefix PTE tag $C;           /* KERNEL OWNER
    constant EOWN equals %X00800000 prefix PTE tag $C;  /* EXEC OWNER
    constant SOWN equals %X01000000 prefix PTE tag $C;  /* SUPER OWNER
    constant UOWN equals %X01800000 prefix PTE tag $C;  /* USER OWNER
end PTEDEF;

end_module $PTEDEF;
```

```
module $PTRDEF;

/*+
/* POINTER CONTROL BLOCK
/*          THIS IS A STRUCTURE OF POINTERS TO OTHER DYNAMIC STRUCTURES
/*          OF LIKE KIND.  TYPICALLY THE STRUCTURES POINTED TO ARE KNOWN
/*          BY THEIR LONG WORD INDEX INTO THE TABLE AND TO FACILITATE FETCHING
/*          THESE, IT IS CONVENTIONAL TO KEEP A POINTER TO THE BASE OF THE
/*          STRUCTURE POINTERS RATHER THAN (OR IN ADDITION TO) THE POINTER
/*          TO THE FRONT OF THE POINTER CONTROL BLOCK.  THE NUMBER OF POINTERS
/*          IN THE ARRAY PRECEEDS THE FIRST POINTER IN THE ARRAY.
/*-


aggregate PTRDEF structure prefix PTR$;
    FILL_1 quadword fill prefix PTRDEF tag $$;          /*RESERVED QUAD WORD FOR LINKAGE
    SIZE word unsigned;                                 /*SIZE OF DYNAMIC CONTROL BLOCK
    TYPE byte unsigned;                                 /*TYPE OF DYNAMIC CONTROL BLOCK
    PTRTYPE byte unsigned;                              /*TYPE OF CONTROL BLOCK POINTED TO
    PTRCNT longword unsigned;                           /*COUNT OF ENTRIES
    constant "LENGTH" equals . prefix PTR$ tag K;       /*LENGTH OF FIXED PORTION
    constant "LENGTH" equals . prefix PTR$ tag C;       /*LENGTH OF FIXED PORTION
    PTR0 longword unsigned;                             /*PTR NUMBER 0

end PTRDEF;

end_module $PTRDEF;
```

0371 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



SYSDEFMP
SDL

SYSDEFQZ
SDL